The Thesis committee for Henri Christian Kjellberg
Certifies that this is the approved version of the following thesis:

Design of a CubeSat Guidance, Navigation, and Control Module

APPROVED BY

SUPERVISING COMMITTEE:

E. Glenn Lightsey, Supervisor

Wallace Fowler

Design of a CubeSat Guidance, Navigation, and Control Module

by

Henri Christian Kjellberg, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

To Kristen and Ender, for keeping me sane.

Acknowledgments

Dax Garner $\boxed{\star}$, for being my companion throughout our time at University. Our conversations echo in my mind still helping me navigate through life to find truth, balance, and happiness.

Dr. E. Glenn Lightsey, for trusting us students and running the Satellite Design Laboratory with the perfect amount of guidance and freedom.

Travis Imken, Kit Kennedy, Katharine Brumbaugh, and all students (affectionately known as burlaks) of the Satellite Design Laboratory for working tirelessly pulling our spacecraft closer to orbit.

Jahshan Bhatti, for guiding me through the dark forest of the written qualifying exams and easily fixing all the very hard problems that I encounter.

Ron Maglothin, for his leadership and tenacity that put our first spacecraft into orbit.

Sebastian Muñoz and Dr. John Christian, for teaching by example.

Design of a CubeSat Guidance, Navigation, and Control Module

Henri Christian Kjellberg, M.S.E.
The University of Texas at Austin, 2011

Supervisor: E. Glenn Lightsey

A guidance, navigation, and control (GN&C) module is being designed and fabricated as part of a series of CubeSats being built by the Satellite Design Laboratory at the University of Texas. A spacecraft attitude control simulation environment called StarBox was created in order to perform trade studies and conduct performance analysis for the GN&C module. Navigation and control algorithms were tested using StarBox and then implemented onto an embedded flight computer. These algorithms were then tested in a hardware-in-the-loop simulation. In addition, the feasibility of utilizing advanced constrained attitude control algorithms was investigated by focusing on implementation in flight software. A mechanical and electrical design for the GN&C module was completed. A prototype system was set up on a bench-top for integrated testing. The analysis indicates that the system will satisfy the requirements of several CubeSat missions, including the current missions at the University of Texas known as Bevo2 and ARMADILLO.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Work on advanced CubeSat missions is paving the way for picosatellites to satisfy requirements for commercial, national security, and research missions. Programs such as NASA Ames's PharmaSat[1] and the National Science Foundation's "CubeSat-based Science Missions for Space Weather and Atmospheric Research"[2] underscore the notion that CubeSats are increasingly being considered viable platforms for scientific research. However, many measurement and communication payloads require pointing or orbital maneuvers. Thus, missions that can be performed by picosatellites are limited by the lack of advanced, miniaturized six degree-of-freedom guidance, navigation, and control (GN&C) systems. Development of a bolt-on, autonomous GN&C module for CubeSats is necessary in order to enable increasingly more complex missions to be conducted in the CubeSat form factor. This chapter introduces the CubeSat standard, and provides an overview of the contributions and contents of this thesis.

1.1 The CubeSat Standard

The CubeSat standard developed by California Polytechnic State University has enabled a new class of low cost missions.[3] Roughly speaking, a 1-U CubeSat is a 10 *cm* linear dimension cubic satellite, with a mass of 1.333 kg. These Cube-

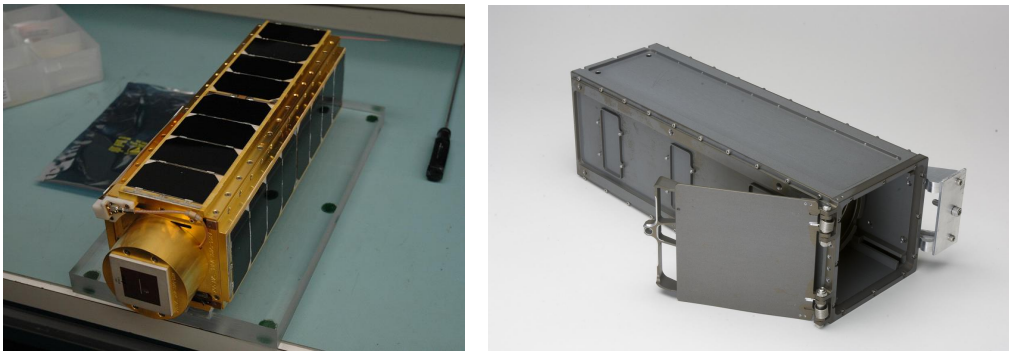


Figure 1.1.1: NASA's PharmaSat (left) and a P-POD (right). [Photo Credit: Christopher Beasley][4]

Sats are launched from a standardized deployment mechanism known as the Poly Picosatellite Orbital Deployer (P-POD). The P-POD can deploy three 1-U CubeSats or one 3-U CubeSat with a total volume of approximately 10 cm x 10 cm x 34 cm and a maximum allowable mass of 4 kg. Developing a spacecraft to the CubeSat standard simplifies the process of obtaining a launch because the P-POD provides a reliable and safe interface to launch vehicles. In general, launch providers are willing to include CubeSats as secondary payloads because the CubeSats are stowed in fully enclosed containers, thus reducing the risk to the primary payload. A typical science grade 3-U CubeSat and its deployment mechanism are shown in Figure 1.1.1.

1.2 Contributions

The goal of this thesis is to guide the reader through the design and development process of a picosatellite GN&C module. In particular, the module is intended to satisfy the requirements of many potential CubeSat missions, including

the NASA sponsored Bevo-2 and Air Force sponsored ARMADILLO CubeSats which are currently under development at the University of Texas at Austin (UT-Austin) Satellite Design Laboratory (SDL). The goal of this research is to create an end-to-end design for a 6 degree-of-freedom GN&C module that can be used on a wide range of CubeSat missions. This is accomplished by creating a simulation environment, performing hardware trade studies, developing navigation and control algorithms, and investigating advanced constrained control algorithms.

1.3 Thesis Organization

This thesis has a total of 8 Chapters. Chapter 2 provides motivation by surveying the satellite projects at UT-Austin's Satellite Design Laboratory. From these representative spacecraft missions, a set of performance requirements are developed for the GN&C module. Chapter 3 describes a spacecraft simulation and analysis tool named StarBox that was created in order to simplify the design process. Chapter 4 focuses on trade studies of commercial-off-the-shelf (COTS) GN&C modules as well as sensors, actuators, and embedded computers that are needed to create the new GN&C module. Chapter 5 describes navigation and control algorithms that compose the majority of the GN&C module flight software. These algorithms are implemented and evaluated using StarBox. Chapter 6 surveys advanced constrained attitude control algorithms in the literature and shows an implementation on an embedded computer. Chapter 7 shows the mechanical and electrical implementation of the GN&C module and hardware-in-the-loop simulation using StarBox. Chapter 8 concludes the thesis with thoughts and reflections on the design process.

Chapter 2

Motivation

Picosatellites have the potential to revolutionize the way spacecraft missions are operated. The National Science Foundation is funding numerous CubeSats to conduct space weather research.[2] However, the current set of missions do not require active attitude control because reliable picosatellite attitude control has not yet been demonstrated to the satisfaction of the sponsors. A science requirement for attitude control is viewed as an unacceptable risk for a low cost CubeSat mission.

A robust attitude control system architecture is needed so that the scope of possible CubeSat missions can be increased to include missions with realistic attitude control requirements and constraints. Developing maneuverable picosatellites is a technical challenge that must be overcome in order to perform coordinated tasks such as formation flying and automated rendezvous and docking. In fact, NASA awarded a Small business Technology TRansfer (STTR) grant to a local start-up company known as Austin Satellite Design (ASD) for their proposal "Guidance, Navigation, and Control System for Maneuverable Pico-Satellites."[5] ASD is working with the UT-Austin SDL for the development of a 6-degree of freedom CubeSat GN&C module. A compact, low power GN&C system will be fabricated and tested for use on CubeSats. This thesis focuses on the development of a CubeSat guidance, navigation, and control module (GN&C) which can support

a wide range of future missions with basic attitude and orbit control requirements.

This chapter introduces the Satellite Design Laboratory, its past spacecraft missions, and current spacecraft missions.

2.1 Satellite Design Laboratory

Students at the UT-Austin SDL design, fabricate, test, and operate small spacecraft. As of 2011, the SDL at UT-Austin has already developed and launched three student designed and student built satellites through research programs supported by NASA and the United States Air Force. Currently, two additional spacecraft are under development. The SDL is also proposing additional CubeSats for interesting scientific surveys in space weather research.

2.2 Past Missions

2.2.1 FASTRAC

AFRL sponsors a student-built spacecraft competition called the University NanoSat Program (UNP). The UNP competition is held every other year. Twelve universities compete for the opportunity to fly their small spacecraft. The SDL developed the winning mission consisting of two nanosatellites (spacecraft that weigh less than 100 kg), for the UNP-3 competition in 2005. This mission was known as FASTRAC (Formation Autonomy Spacecraft with Thrust, Relnav, Attitude, and Crosslink) and it was designed to demonstrate technologies related to satellite formation flight and GPS relative navigation.[6] Figure 2.2.1 shows the FASTRAC spacecraft in the foreground on-top of the STP-S26 Minotaur launch vehicle pay-

Figure 2.2.1: FASTRAC shown in the foreground.[7]

load adapter plate. The satellites were successfully launched into orbit on-board STP-S26 in Fall 2010.

While FASTRAC has been an outstanding success, it took 7 years to design, develop, manifest, and launch the spacecraft. Most of that time was spent waiting for a launch opportunity. Nanosatellites are simply more difficult to manifest on a suitable launch vehicle due to their larger size. [6] FASTRAC continues to operate successfully as of the Summer 2011.

2.2.2 LONESTAR Mission 1: Bevo-1

In comparison, the SDL student designed and built picosatellite known as Bevo-1 was able to be launched into orbit in less than 2 years of total time. Bevo-1

was the first in a series of four missions outlined in the NASA Johnson Space Center LONESTAR (Low Earth Orbiting Navigation Experiment for Spacecraft Testing Autonomous Rendezvous and docking) Program.[8] The goal of this program is to demonstrate an autonomous rendezvous and docking capability between two cooperative small spacecraft by the fourth mission. For the first mission, UT-Austin and Texas A&M University each independently developed a spacecraft with the relatively simple goal of testing a NASA developed GPS receiver.

The Space Shuttle Picosatellite Launcher (SSPL) is similar to the P-POD in that it provides a fully enclosed deployment mechanism that can house multiple small spacecraft. Instead of storing three 1-U CubeSats, the SSPL allows for two 5 *in* cubic spacecraft. Figure 2.2.2 shows Bevo-1, its partner spacecraft the AggieSat-2, and the SSPL. Figure 2.2.3 show the two LONESTAR-1 spacecraft as they were deployed from the Space Shuttle cargo bay in July, 15 2009.[8]

2.3 Current Missions

2.3.1 LONESTAR Mission 2: Bevo-2

UT-Austin is currently working on a 3-Unit CubeSat called Bevo-2 in support of the LONESTAR program's second mission. The mission objectives for Bevo-2 are to demonstrate technologies necessary for autonomous rendezvous and docking.[9] Specifically, Bevo-2 shall demonstrate a six degree-of-freedom GN&C system. This system includes both an attitude determination and control system as well as a small cold gas thruster for changing orbital parameters. The algorithms that operate the sensor and actuator suite are being developed so that they will be easily utilized by a host satellite. Figure 2.3.1 shows a computer model of the cur-

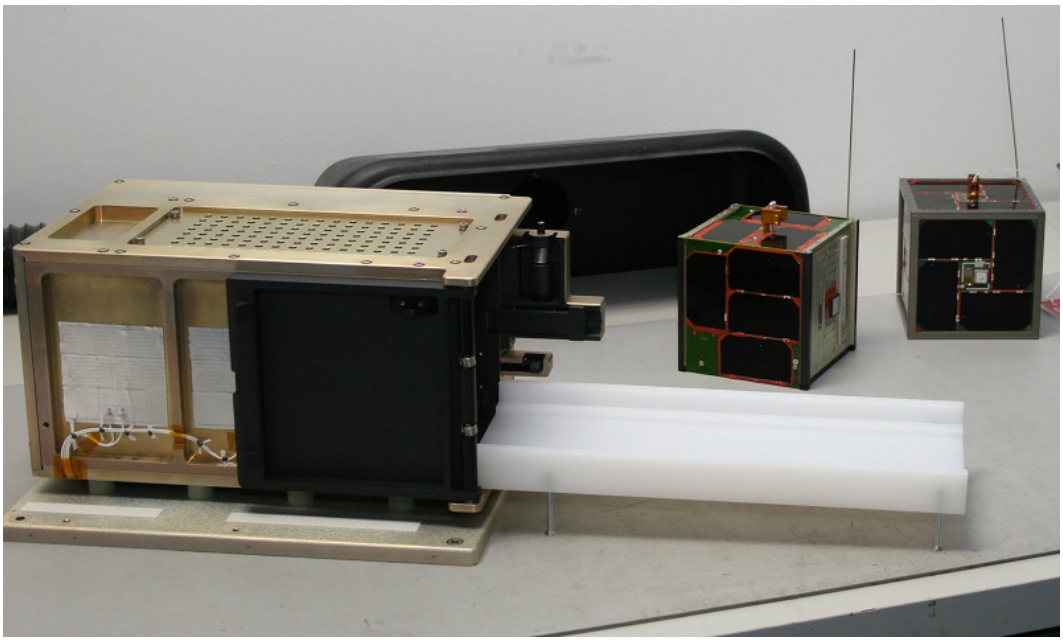


Figure 2.2.2: The LONESTAR-1 system consisting of the Space Shuttle Picosatellite Launcher (left), Bevo-1 (middle), and AggieSat-2 (right). [Photo Credit: Henri Kjellberg]

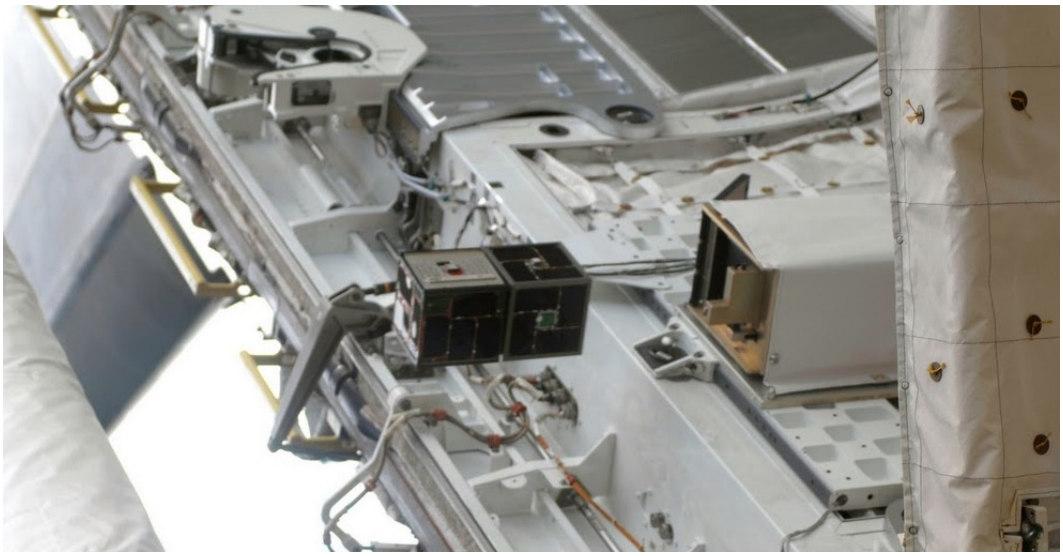


Figure 2.2.3: The Bevo-1 and AggieSat-2 leaving the SSPL on STS-127 Endeavour. [Photo Credit: NASA]

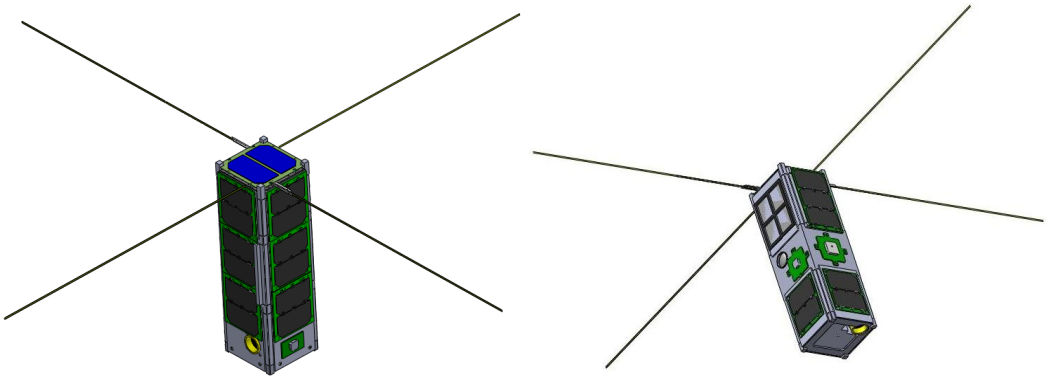


Figure 2.3.1: The Bevo-2 spacecraft (left) and the UNP-7 entry ARMADILLO (right).

rent design of Bevo-2.

Bevo-2 will perform a cooperative mission with the Texas A&M University satellite known as AggieSat-4. AggieSat-4 is an approximately 50 kg nanosatellite that will stow Bevo-2 using a P-POD-like deployment mechanism. The two spacecraft will be launched into orbit aboard an International Space Station (ISS) resupply mission. From the ISS, the two spacecraft will be deployed into orbit using the Japanese Aerospace Exploration Agency (JAXA) nanosatellite launcher. The two spacecraft will perform crosslink communications right after separation. After initial checkout, there is a sensor, actuator, and propulsion test. Once the GN&C module has been characterized, a series of maneuvers will be performed with a chosen state in space. Figure 2.3.2 shows a cartoon diagram of the concept of operations for Bevo-2.

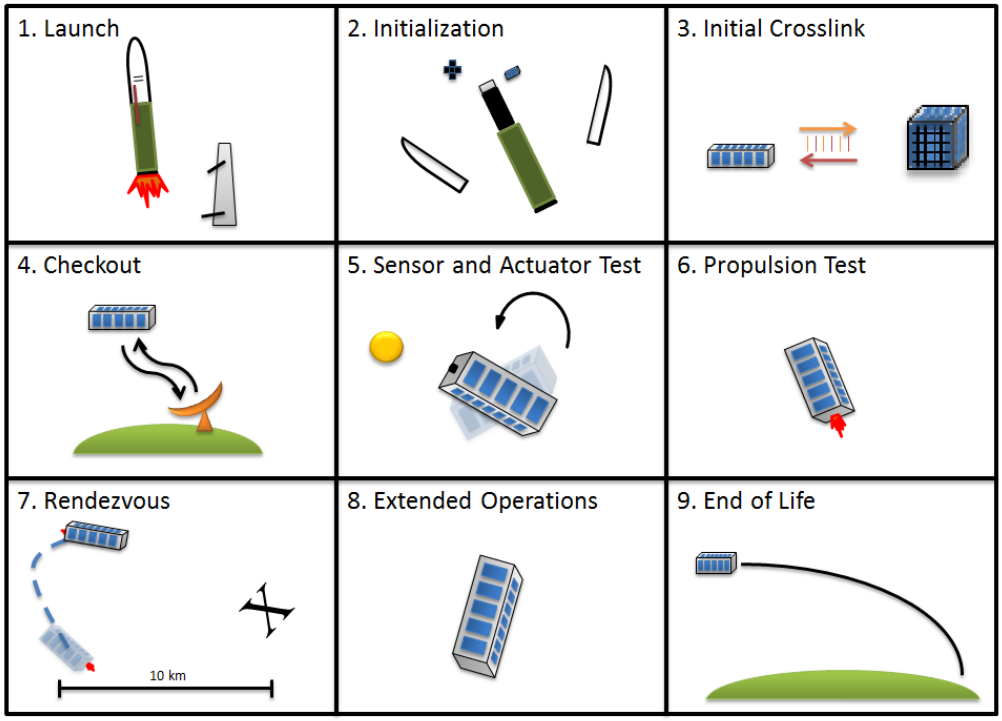


Figure 2.3.2: Concept of operations for Bevo-2 for the LONESTAR-2 mission.[9]

2.3.2 ARMADILLO

UT-Austin's current entry into the UNP-7 competition (2011-2013) is known as ARMADILLO (Attitude Related Maneuvers And Debris Instrument sateLlite in Low Orbit). ARMADILLO is a 3-U CubeSat that will perform measurements of sub-millimeter size space dust in low Earth orbit using a piezoelectric dust detector being developed by Baylor University.[10] Figure 2.3.1 shows the ARMADILLO spacecraft, which has a similar spacecraft bus design as Bevo-2. In fact, the GN&C module is identical for the two spacecraft. This design approach reduces the engineering costs associated with the module. Figure 2.3.3 shows a cartoon diagram of

the concept of operations for ARMADILLO. The science payload requires that the piezoelectric dust detector be pointed along the spacecraft velocity vector within 5 degrees. Pointing is also required for high bandwidth data communications with ground stations. A cold gas thruster is utilized to increase the lifetime of the spacecraft in the low altitude orbit.

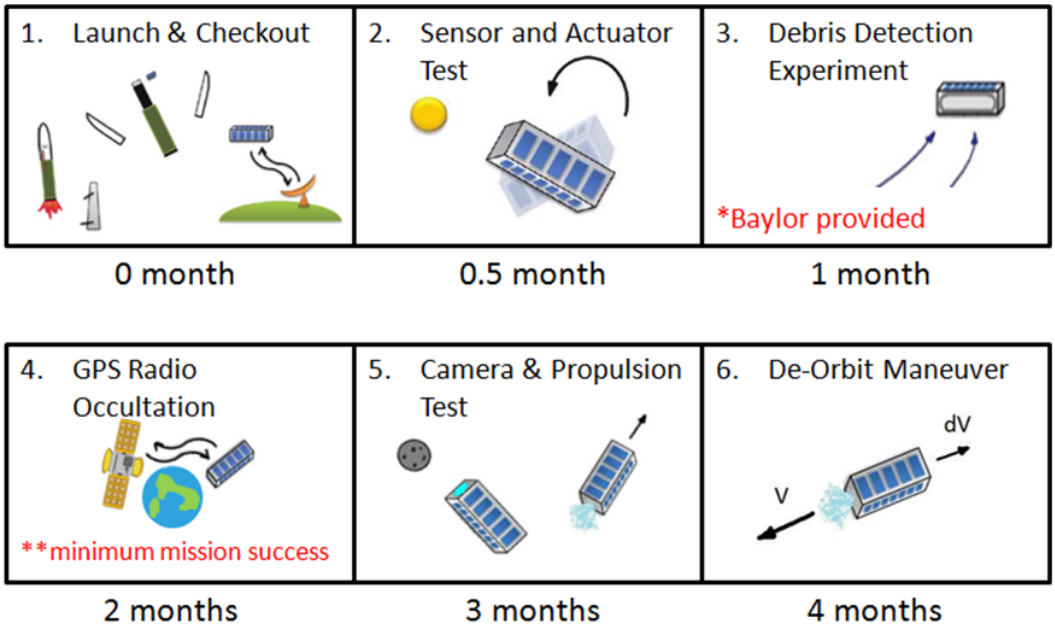


Figure 2.3.3: Concept of operations the ARMADILLO CubeSat.[10]

2.3.3 GN&C Module Requirements

With the Bevo-2 and ARMADILLO CubeSats in mind, a common set of performance requirements were developed that will satisfy both missions.

- The GN&C module shall occupy less than $1/3$ of the 3-U CubeSat volume.
  - Rationale: While the services provided by the GN&C module are com-

plex, the most important part of any mission is the actual payload. Allowing 1-U for the payload, 1-U for the GN&C module, and 1-U for the remainder of the spacecraft bus is considered to be an acceptable distribution of the available volume.

- The GN&C module shall provide less than 5 degree pointing error (1σ) over 1 full orbit with a 50% duty cycle.
  - Rationale: For the ARMADILLO mission, the scientific measurements need to be acquired while pointing one face of the spacecraft within 5 degrees of the velocity vector. The longer this attitude is held, the more measurements can be obtained. A 50% duty cycle allows for at least half the spacecraft's orbit time to be dedicated to obtaining science measurements. The other 50% can be utilized to perform spacecraft maintenance functions, such as ground communications, momentum management, and battery charging.
- The GN&C module shall perform reorientation maneuvers from an arbitrary direction to any target direction in less than 1 minute.
  - Rationale: For Bevo-2, the rendezvous maneuvers require small correction thrusts throughout the maneuver. Subsequent mid-course maneuvers can occur with as little as 2 minutes apart time separation. Attitude maneuverability is needed to be able to point the thruster nozzle in the correct direction for orbit maneuvers.

Bevo-2 and ARMADILLO are expected to be launched into orbit on-board ISS resupply missions. Therefore, the design orbit for the GN&C module assumes a low Earth orbit with an altitude of 400 km.

Chapter 3

The StarBox Spacecraft Simulator

In order to design a GN&C module for a CubeSat, it is necessary to first understand the principles of the attitude determination and control problem. One way to learn about this problem is to create a software spacecraft simulator. The StarBox spacecraft simulator was designed as a tool to help in the development of the GN&C module. StarBox simulates a full 6 degree-of-freedom spacecraft in an environment with the imperfections of perturbation torques, and sensor and actuator noise. Specifically, StarBox was created to assist in choosing the hardware components and the development of the filter and control algorithms. StarBox allows for the performance of the GN&C module to be simulated with the hardware and algorithm components working together as a system. StarBox is intended to be a modular simulation environment in order to allow for different environment and hardware models to be plugged in and out of the simulation. The realism of the simulations can be adjusted by disabling or enabling various error sources and perturbations.

This chapter presents the spacecraft dynamics model, sensor and actuator models, disturbance models, and provides an overview of the implementation of the StarBox simulator. Figure 3.0.1 shows a block diagram of the modules that constitute the StarBox spacecraft simulator.

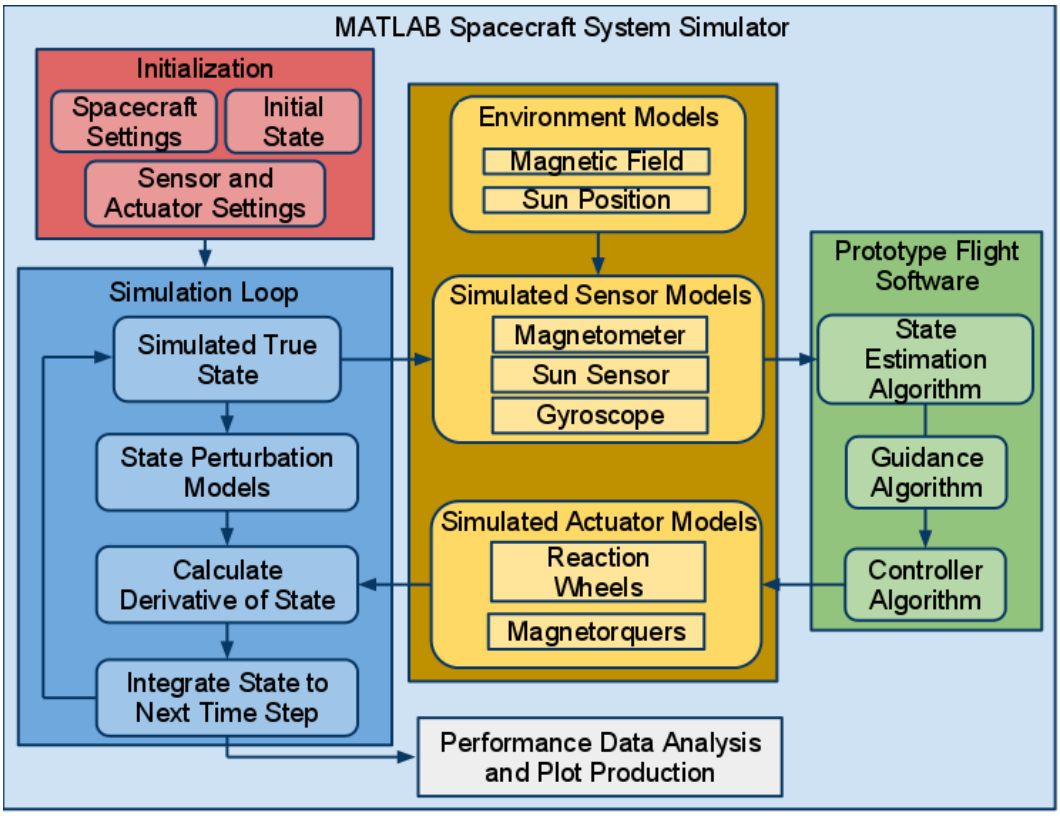


Figure 3.0.1: Block diagram describing the interconnections between each module of the StarBox spacecraft simulator.

3.1 Spacecraft Dynamics Model

StarBox accepts a spacecraft of any form factor and size. For this analysis all the work was done using a 3-U CubeSat modeled as a uniform rectangular prism with dimensions of $100 \times 100 \times 340$ mm and a total mass of 4 kg. The center of mass of the spacecraft is then offset by 2 cm in order to simulate the worst case scenario center-of-mass and center-of-pressure offset. The 2 cm offset is the maximum allowed in the CubeSat specifications.[3] It is assumed that there are no deployable or flexible structures; therefore Euler's equations of a rigid body motion are assumed to apply.

The spacecraft dynamic state is defined as,

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ \bar{\mathbf{q}} \\ \boldsymbol{\omega} \\ \mathbf{h}_w \end{bmatrix}_{17\times 1} . \tag{3.1.1}$$

Here, $\mathbf{r}$ and $\mathbf{v}$ are the position and velocity of the spacecraft center of mass in the Earth centered inertial (ECI) frame, $\bar{\mathbf{q}} = \begin{bmatrix} \mathbf{q} & q \end{bmatrix}$ is the attitude quaternion where the first component is the vector and the second component is the scalar component, $\boldsymbol{\omega}$ is the body rate, and $\mathbf{h}_w$ is the current total angular momentum that is stored in the reaction wheels, that is $\mathbf{h}_w = \sum_{i=0}^{k} \mathbf{h}_{w_i}$ for all k reaction wheels. The translational acceleration of the spacecraft center of mass is described as,

$$\dot{\mathbf{v}} = -\frac{\mu}{|\mathbf{r}|^3}\mathbf{r} + \mathbf{F}_D/m. \tag{3.1.2}$$

Here, μ is the Earth's gravitation parameter, $\mathbf{F}_D$ is the force of drag, and m is the mass of the spacecraft. Orbital perturbations such as J2 are not incorporated because StarBox is predominantly meant to analyze the attitude control performance and these effects are higher order. The rate of change of the quaternion is given using

$$\dot{\mathbf{q}} = \begin{bmatrix} \frac{1}{2}q\boldsymbol{\omega} - \boldsymbol{\omega} \times \mathbf{q} \\ -\frac{1}{2}\boldsymbol{\omega}^T \mathbf{q} \end{bmatrix}. \tag{3.1.3}$$

The rate of change of the angular velocity is given by

$$\dot{\boldsymbol{\omega}} = J^{-1}\left([\mathbf{h}_s\times]\,\boldsymbol{\omega} + \boldsymbol{\tau}_a + \boldsymbol{\tau}_m + \boldsymbol{\tau}_M + \boldsymbol{\tau}_A\right). \tag{3.1.4}$$

Here, $\boldsymbol{\tau}_a$ and $\boldsymbol{\tau}_m$ indicate the control torques from the reaction wheels and magnetorquers respectively. $\boldsymbol{\tau}_M$ and$\boldsymbol{\tau}_A$ indicate disturbance torques from aerodynamic drag and the residual magnetic dipole. The $[\mathbf{a}\times]$ operator indicates the vector $\mathbf{a}$ is used to create a skew symmetric matrix,

$$[\mathbf{a}\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{3.1.5}$$

$\mathbf{h}_s$ is the total angular momentum of the spacecraft, including the components from the reaction wheels,

$$\mathbf{h}_s = \mathbf{J}\boldsymbol{\omega} + \mathbf{h}_w. \tag{3.1.6}$$

These equations incorporate the contribution of stored momentum from the reaction wheels into the dynamics of the system in a manner similar to the method used by Tsiotras.[11]

3.2 Sensor and Actuator Models

Each sensor and actuator model attempts to model the imperfections in the measurements or control torques. These imperfections are important because they can affect the attitude estimate, control solutions, and the dynamics of the spacecraft. A control algorithm that works perfectly with a perfect attitude estimate may produce unacceptable performance with an imperfect attitude estimate. This section provides a description of the sensor and actuator models that are used within StarBox. These sensor and actuator types were selected because they are used in the CubeSat GN&C module that is being developed for Bevo-2 and ARMADILLO.

3.2.1 Sun Sensor and Sun Position Models

For StarBox, a simple model for the sun sensor is implemented. For two-axis digital sun sensors, it was assumed that the azimuth and elevation produce independent errors in the measurements in the body frame. The implementation follows a similar method to Muñoz.[12]

Step 1: The position of the sun in the inertial frame is obtained using the JPL SPICE library. Using the true components of the unit vector of the sun from the sun sensor $\mathbf{r}_{\odot/ss} = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}$, StarBox computes the true elevation

and azimuth respectively,

$$El_t = \arcsin[r_3] \tag{3.2.1}$$

$$Az_t = \arctan\left[\frac{r_2}{r_1}\right]. \tag{3.2.2}$$

Step 2: StarBox then corrupts the true angles with bias and noise using information from the hardware data sheets or from testing to obtain the measured elevation and azimuth,

$$\begin{bmatrix} El_m \\ Az_m \end{bmatrix} = \begin{bmatrix} El_t \\ Az_t \end{bmatrix} + \mathbf{b} + \mathbf{n}_{ss}. \tag{3.2.3}$$

Here, $\mathbf{b}$ denotes a random, permanent bias, $\mathbf{n}_{ss}$ denotes a white noise random process with constant standard deviation.

Step 3: StarBox then forms the vector measurement of the sun direction in the sun sensor reference frame,

$$\mathbf{e}'_{\odot/ss} = \begin{bmatrix} \cos El_m \cos Az_m \\ \cos El_m \sin Az_m \\ \sin El_m \end{bmatrix}. \tag{3.2.4}$$

Step 4: Finally, StarBox adds a small misalignment error rotation matrix $\mathbf{\Gamma}_{ss}$. For a small angle $\delta\psi$ misalignment about a randomly chosen axis $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T$, the rotation can be expressed as,

$$\mathbf{\Gamma}_{ss} \approx \mathbf{I}_{3x3} - [\mathbf{u}\delta\psi\times] = \begin{bmatrix} 1 & u_3\delta\psi & -u_2\delta\psi \\ -u_3\delta\psi & 1 & u_1\delta\psi \\ u_2\delta\psi & -u_1\delta\psi & 1 \end{bmatrix}. \tag{3.2.5}$$

Using the misalignment error rotation matrix, the final measurement becomes

$$\mathbf{e}_{\odot/ss} = [\mathbf{I}_{3\times3} + \mathbf{\Gamma}_{ss}]\,\mathbf{e}'_{\odot.ss}. \tag{3.2.6}$$

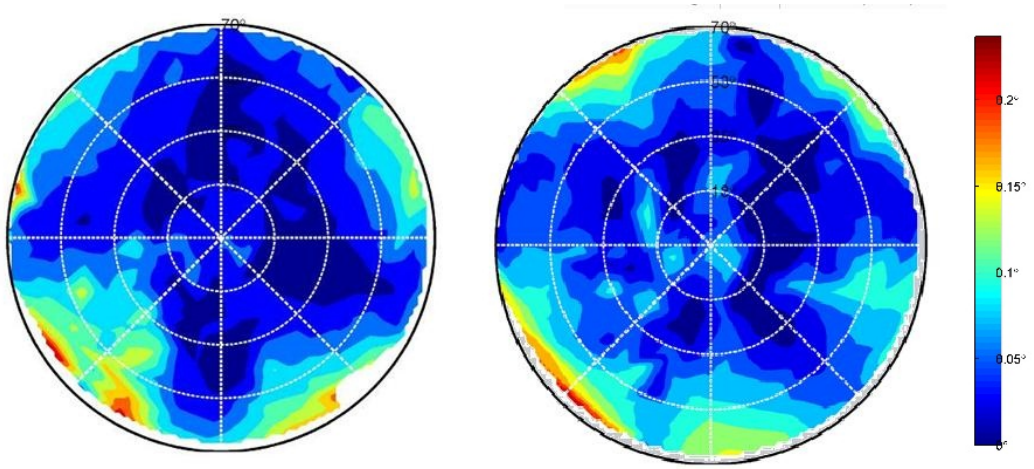


Figure 3.2.1: Heat map of the boresight vector angle error in the sun sensor reference frame as a function of position in the field of view of two sun sensors of the same model. The errors range from 0 (dark blue) to 0.25 degrees (dark red).[13]

Indeed, the truth is far more complicated as can be seen in actual sun sensor testing data from Sinclair Interplanetary shown in Figure 3.2.1. Note that individual sun sensors of the same type can have different error distributions. Sinclair demonstrates their methods of testing the measurement error in the SS-411 sun sensors.[14] At the current stage of the simulation, it is not important to get the measurement error sources perfect. However, due to the modular nature of the StarBox simulation environment, an empirical model such as a table look-up could be placed in to evaluate the performance characteristics with increasing fidelity.

3.2.2 Magnetometer and Magnetic Field Models

The magnetometer has two main sources of error as an attitude sensor: these are measurement error and model error. The magnetometer's performance characteristics are about as good as the performance of the magnetic field reference model.

The magnetic field model being used in StarBox is the World Magnetic Model for 2010-2015. The WMM2010 model is indicated as corresponding with truth to approximately 1 degree in inclination, declination, and grid variation in 2010.[15]

In order to produce a simulated measurement, StarBox must include the error between the magnetic field model that will be flown on board the spacecraft and the actual magnetic field. However, in StarBox, both the true magnetic field and the measured magnetic field come from the same model. Instead, StarBox treats the error from the magnetic field model as an additional misalignment error. The process for calculating the magnetic field measurements in StarBox is outlined below.

Step 1: StarBox finds the magnetic field vector in the inertial frame, $\left(\boldsymbol{\beta}'_{b/wmm}\right)_I$ from the WMM2010 model using a 12×12 spherical harmonic expansion. This is the spacecraft's on-board reference field direction.

Step 2: StarBox then rotates the model magnetic field model in a random direction by 1 degree. The result, $\left(\boldsymbol{\beta}'_{b/t}\right)_I$ is considered the true magnetic field direction.

$$\left(\boldsymbol{\beta}'_{b/t}\right)_I = (\mathbf{I}_{3\times 3} + \boldsymbol{\Gamma}_{wmm}) \left(\boldsymbol{\beta}'_{b/wmm}\right)_I \tag{3.2.7}$$

Here, $\boldsymbol{\Gamma}_{wmm}$ is the misalignment error from the magnetic field model randomly chosen for each measurement.

Step 3: StarBox now rotates the true magnetic field vector into the magnetometer frame and adds noise and bias to produce the magnetic field measurement

produced by the magnetometer prior to misalignment,

$$\left(\boldsymbol{\beta}'_{b/t}\right)_M = \mathbf{T}_M^I \left(\boldsymbol{\beta}'_{b/t}\right)_I \tag{3.2.8}$$

$$\left(\boldsymbol{\beta}'_{b/m}\right)_M = \left(\boldsymbol{\beta}'_{b/t}\right)_M + \mathbf{b}_{mag} + \mathbf{n}_{mag}. \tag{3.2.9}$$

Here, $\mathbf{b}_{mag}$ is the bias of the magnetometer and $\mathbf{n}_{mag}$is white noise.

Step 4: Finally the measurement is rotated again to account for misalignment of the sensor

$$\left(\boldsymbol{\beta}_{b/m}\right)_M = \left[\mathbf{I}_{3\times 3} + \boldsymbol{\Gamma}_{mag}\right] \left(\boldsymbol{\beta}'_{b/m}\right)_M. \tag{3.2.10}$$

Here, $\boldsymbol{\Gamma}_{mag}$ is chosen at random at the beginning of the simulation, and remains constant throughout the simulation.

3.2.3 Gyroscope Model

The gyroscope model is slightly more complicated than the sun sensor and magnetometer. In addition to a bias and noise, gyroscopes can also experience a change in their bias. The model is similar to what is used by Christian [16] and Muñoz [12]. Here the bias drift is derived in a slightly more explicit manner. The process to calculate a simulated gyroscope measurement is as follows:

Step 1: The bias is updated with time based on its constant in-run bias stability, $\dot{\mathbf{b}}_g$, and a random zero mean noise $\boldsymbol{\eta}$,

$$\mathbf{b}_g^+ = \mathbf{b}_g^- + \dot{\mathbf{b}}_g \boldsymbol{\eta} \Delta t. \tag{3.2.11}$$

Step 2: The rest of the model is constructed similarly to the magnetometer model,

$$\boldsymbol{\omega}_m = \left[\mathbf{I}_{3\times 3} + \boldsymbol{\Gamma}_g\right] \left(\boldsymbol{\omega}_t + \boldsymbol{b}_g^+ + \mathbf{n}_g\right). \tag{3.2.12}$$

Here, $\mathbf{\Gamma}_g$ is the misalignment error matrix, $\boldsymbol{\omega}_t$ is the true angular velocity of the spacecraft, $\mathbf{b}_g^+$ is the bias of the gyroscope at the current time, and $\mathbf{n}_g$ is the white noise applied to the gyroscope.

3.2.4 Reaction Wheel

The reaction wheel model has two components. The reaction wheel is commanded as a torquing device. The torque produced by the reaction wheel is corrupted by a noise,

$$\boldsymbol{\tau}_{a_w} = \boldsymbol{\tau}_{c_w} + \mathbf{n}_w. \tag{3.2.13}$$

The momentum stored in the reaction wheel changes the dynamics of the spacecraft. Therefore, the momentum stored in each wheel must be tracked as a component of the state vector. The momentum stored in each wheel is related is related to wheel speed directly,

$$h_w = I_w \omega_w. \tag{3.2.14}$$

The derivative of the wheel momentum is the actual torque experienced by the wheel,

$$\dot{h}_w = \tau_a. \tag{3.2.15}$$

The wheel speed is thus updated during the integration step of the simulation loop.

3.2.5 Magnetorquer Model

The magnetorquer model takes the commanded dipole and crosses it with the magnetic field. Because the magnetorquers are low bandwidth devices compared to the reaction wheels, the model does not include noise or bias,

$$\boldsymbol{\tau}_m = \mathbf{m}_c \times \boldsymbol{\beta}_t. \tag{3.2.16}$$

Here, $\mathbf{m}_c$ is magnetic dipole from the magnetorquers and $\boldsymbol{\beta}_t$ is the true magnetic field.

3.3 Disturbance Models

Three sources of disturbance torques were modeled: residual magnetic dipole, aerodynamic drag torque, and gravity gradient.

3.3.1 Residual Magnetic Dipole

Spacecraft have a non zero magnetic field due to the materials used in construction and the electronics that are operating within the spacecraft. Table 3.3.1 shows a NASA suggested approximation for the residual magnetic dipole of a spacecraft based on the spacecraft's mass and the magnetic property control procedures utilized in its fabrication. The class indicates the level of control that has been exercised during the design process to mitigate the residual dipole of the spacecraft where Class I indicates the largest effort in design, quality control, and testing practices.[17] As a default StarBox assumes Class III levels. This assumption leads the residual magnetic dipole to be $|\mathbf{m}_r| = 0.04\, A \cdot m^2$. The direction of the $\mathbf{m}_r$ vector

Magnetic Properties Control	Dipole Moment per Unit Mass
Class I	$1 \times 10^{-3}\ A \cdot m^2/kg$
Class II	$3.5 \times 10^{-3}\ A \cdot m^2/kg$
Class III	$10 \times 10^{-3}\ A \cdot m^2/kg$

Table 3.3.1: Typical residual magnetic dipole for spacecraft.

is chosen at random at the beginning of each simulation. The torque produced by this residual dipole is calculated as

$$\boldsymbol{\tau}_M = \mathbf{m}_r \times \boldsymbol{\beta}_t. \tag{3.3.1}$$

Here, $\boldsymbol{\tau}_M$ the residual magnetic dipole torque. The true residual magnetic dipole and magnetic field are represented in the inertial frame. The resulting worst case magnetic dipole torque for a CubeSat in the design orbit is approximately $3.2 \cdot 10^{-3}$ mNm.

3.3.2 Aerodynamic Drag Torque

Aerodynamic drag produces both a change in the energy of the spacecraft orbit as well as a torque. The torque exists as long as the aerodynamic drag force vector does not travel through the center of mass of the spacecraft. The CubeSat standard restricts the center of mass deviation to a 2 cm sphere about the geometric center of the spacecraft. The default assumption for StarBox is that the center of mass is fixed at the edge of the 2 cm sphere in order to produce a worst case scenario.[3]

The drag model in StarBox takes into account the spacecraft's orientation with respect to an atmosphere fixed to a rotating Earth as developed by Garner [18].

In order to compute the resultant drag force and drag torques, the following steps are taken:

Step 1: Compute the relative velocity of the spacecraft with respect to the rotating atmosphere.

Step 2: Compute the projection of the total frontal area in the direction of the drag force vector given the spacecraft's current attitude.

Step 3: The atmospheric density is taken from a model such as Jacchia or NRLMSISE-00 for the current spacecraft position.

Step 4: The drag force is then calculated,

$$\mathbf{F}_{drag} = -\frac{1}{2}\rho_{j77}\,|\mathbf{v}_{rel}|\,C_D A_P \mathbf{v}_{rel} \tag{3.3.2}$$

Here, ρ_{j77} is the density from the Jacchia 77 model, $\mathbf{v}_{rel}$ is the velocity of the spacecraft with respect to the rotating atmosphere, C_D is a chosen coefficient of drag, and A_P is the current projected area of the spacecraft.

Step 5: The drag torque is computed from this vector and the location of the center of mass.

$$\boldsymbol{\tau}_{drag} = \mathbf{r}_{F/cm} \times \mathbf{F}_{drag} \tag{3.3.3}$$

The resulting worst-case aerodynamic drag for a CubeSat in the design orbit is approximately $0.5 \cdot 10^{-3}$ mNm.

3.3.3 Gravity Gradient Torque

For completeness, a simple gravity gradient model was implemented, as given by: [19]

$$\mathbf{M} = 3n^2\mathbf{r} \times \mathbf{Jr} \tag{3.3.4}$$

Here, n is the mean motion of the spacecraft, $\mathbf{r}$ is the vector to the center of the earth from the spacecraft in the spacecraft body frame, and $\mathbf{J}$ is the moment of inertia of the spacecraft. However, due to the small mass and the compactness of the CubeSat standard without any deployable structures, the resulting torques are on the order of 10^{-6} mNm and negligible compared to the other environmental disturbance torques.

3.4 Simulation Implementation

The StarBox simulation has been implemented in MATLAB in order to minimize the development time and to increase the number of students who are capable of utilizing the system for their research. The simplicity of programming with MATLAB allows for easy understanding of the operation of the code.

However, MATLAB does not always produce efficient code execution. In the beginning, the entire codebase for StarBox was written in MATLAB. As a result, the execution time, of 100 seconds of simulation time could easily take 500 seconds of real time. This is an unacceptably long period of time to conduct trade studies. In order to reduce the execution time a software execution profiler was employed to identify the bottlenecks in the code. The profiler allows the programmer to view the

execution time and frequency with which each function is called as seen in Figure 3.4.1.

In particular, segments of the code that have high numbers of iterations were reimplemented in C and compiled as a MATLAB Executable (MEX). The MEX files execute far more rapidly as seen in Figure 3.4.2. Here, the MATLAB dot product function has been replaced by a simple MEX equivalent with an execution time reduced by an order of magnitude. Meticulously eliminating the slow functions reduces the execution time. Other parts that were reimplemented as MEX files include the equations of motion and the integration of the time step. As a result of these changes, 100 seconds of simulation time now executes in approximately 40 seconds.

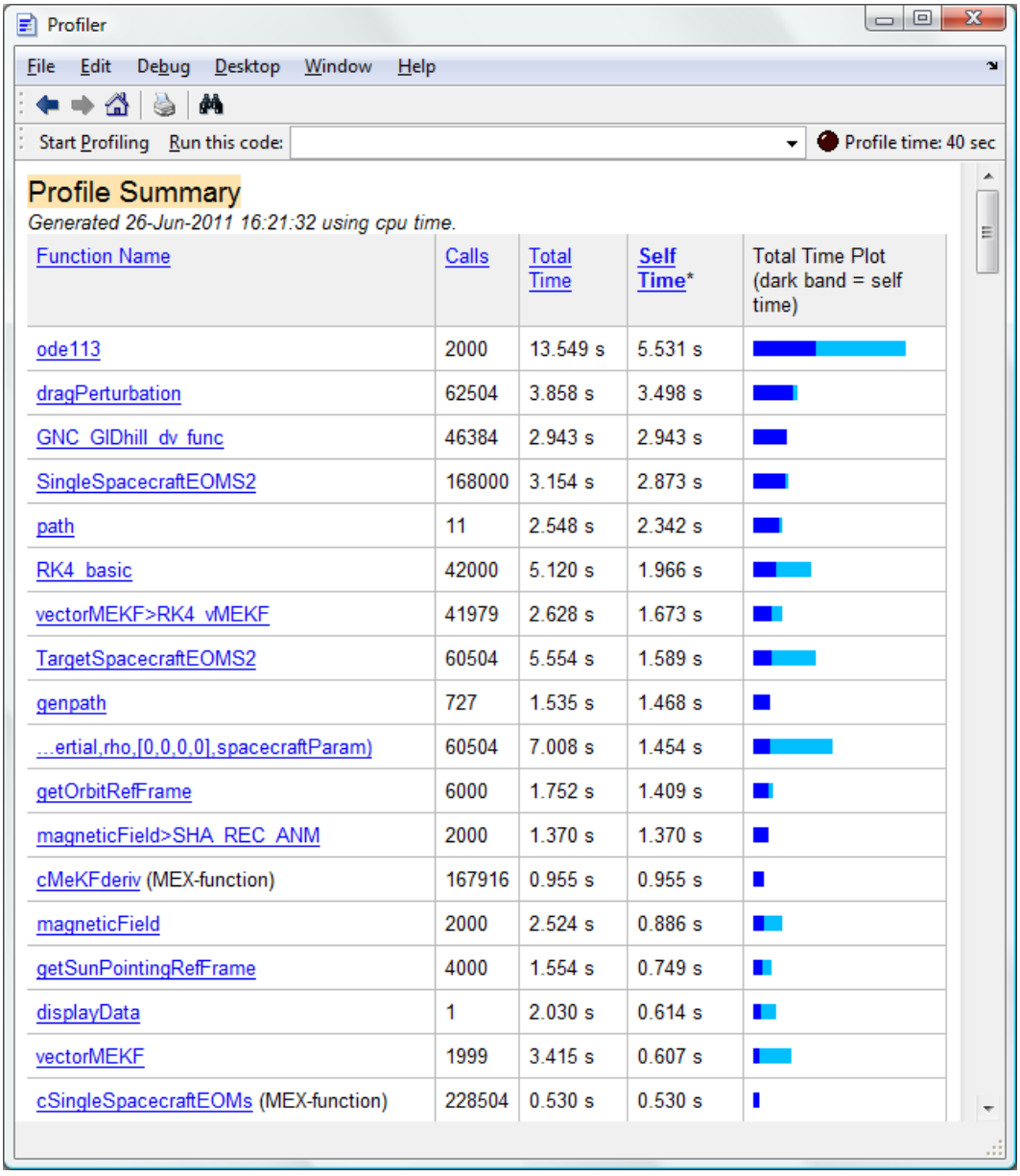


Figure 3.4.1: Profiler used to evaluate the efficiency of each function in StarBox.

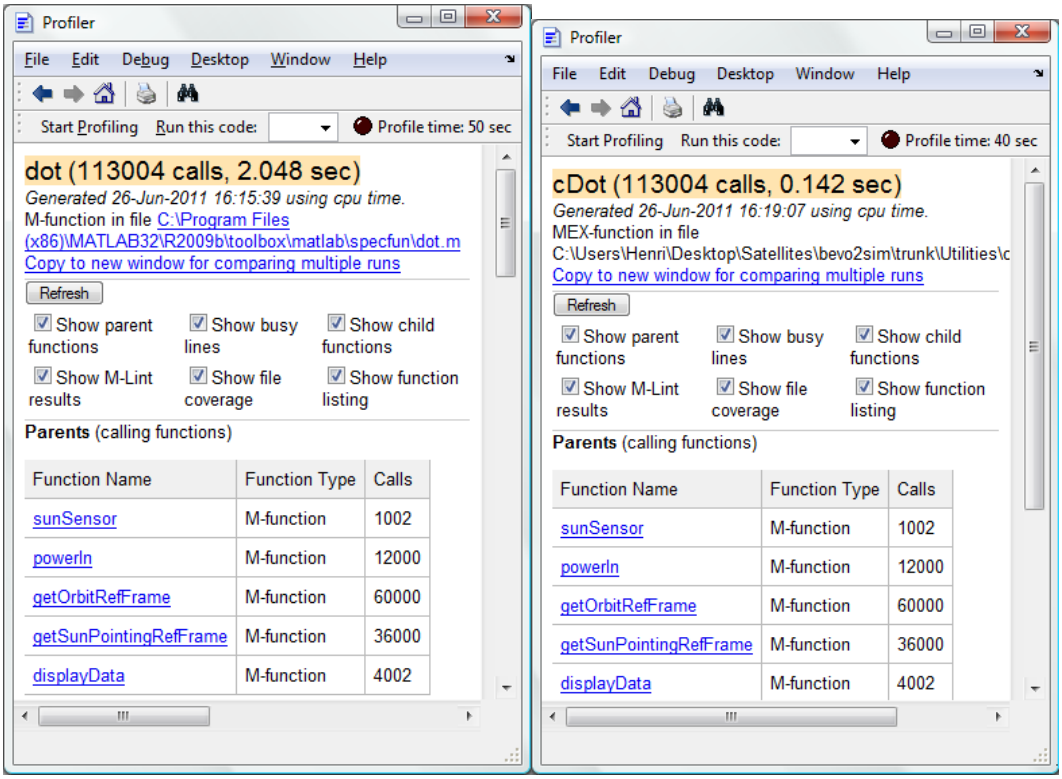


Figure 3.4.2: Computational efficiency of MATLAB dot function (left) vs. MEX file dot function (right).

Chapter 4

Trade Study

The first step in designing a new GN&C system is to investigate the work that has been done in the past for a mission with similar requirements. Through this process one can find whether or not a system already exists that meets the needs of the project. If none of the systems that are identified satisfy the project requirements, then the results of the trade study can be utilized as a basis for creating a component trade study in order to implement the system from the component level. This chapter goes through the a system trade study and a component trade study using the StarBox simulator as an analysis tool.

4.1 System Trade Study

Two commercially available CubeSat GN&C systems were identified: the Maryland Aerospace MAI-100 ADACS and the University of Toronto Institute for Aerospace Studies (UTIAS) CubeSat Compact Three-Axis Attitude Actuator and Sensor Pack. These two systems are shown in Figure 4.1.1.[20, 21] The GN&C modules are very similar in what they provide. Attitude determination is performed by sun sensors and a magnetometer. Attitude control is achieved using a set of reaction wheels for controlling angular momentum and magnetorquer coils in order to unload momentum. An embedded computer performs the necessary calculations

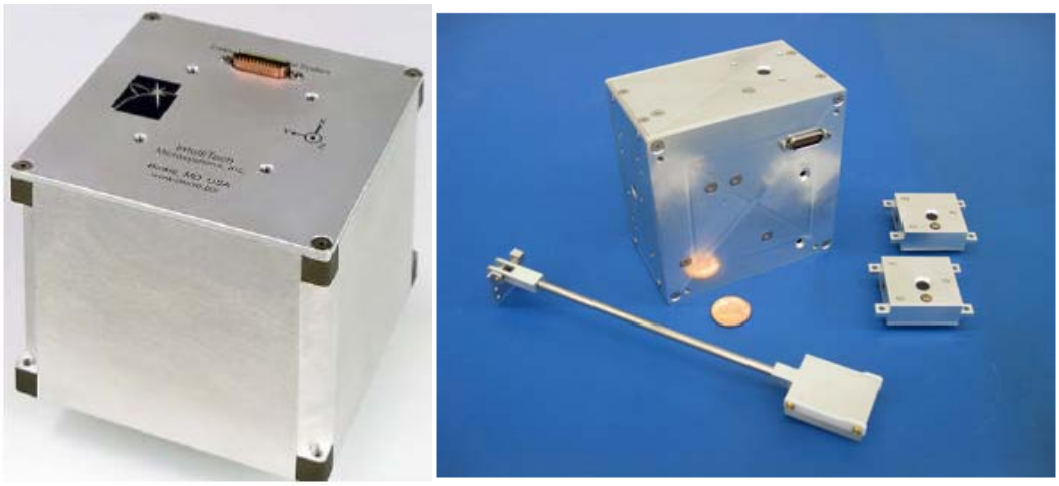


Figure 4.1.1: Commercially available CubeSat attitude determination and control modules, IMI-100 (left) [20] UTIAS (right) [21]

Performance Index	MAI-100	UTIAS
Momentum Storage	1.1 mNm-s	10 mNm-s
Maximum Torque	0.635 mNm	1 mNm
Mass	865 g	< 1000 g
Power	2.4 W	< 1 W
Pointing Accuracy	not provided	1-2 deg RMS

Table 4.1.1: Performance characteristics of the commercial-off-the-shelf CubeSat attitude determination and control modules. [20, 21]

to control the system. Both systems include the actuators and embedded computer inside a bolt-on module, while the some sensors are attached separately throughout the spacecraft. The UTIAS system includes a boom for the magnetometer. Obtaining more information about the systems has been difficult. Table 4.1.1 shows the manufacturer indicated performance characteristics of the modules. The UTIAS system advertised specifications are quite close to the desired performance. However, a smaller GN&C module is desired than what is available from these two commercial sources.

4.2 Component Trade Study

What can be determined from the system trade study is that no systems exist at a mature enough technology readiness level. Both the systems that are available utilize a combination of reaction wheels and magnetorquers to control the attitude of the spacecraft. They utilize sun sensors and magnetometers to perform attitude determination. With this trade space in mind, a component trade study is performed to find devices which satisfy the needs for sensing and actuating the spacecraft state as well an embedded computer to perform the necessary calculations.

4.2.1 Sensors

In order to obtain an inertial attitude estimate at least two linearly independent vector measurements are needed. On small spacecraft, these external measurements can be obtained by taking measurements of the sun direction, measuring the direction of the magnetic field, and measuring star fields. For this module, a sun sensor and magnetometer pair is sought. While a star tracker has the potential to significantly improve attitude estimation performance, no star trackers were available at the time of this writing that fit within the budget of the ARMADILLO and Bevo-2 CubeSat projects.

4.2.1.1 Magnetometers

A spacecraft in the design orbit of approximately 400 km generally experiences a magnetic field with a magnitude of 25000 nT.[17] Many magnetometers exist in the commercial market that can measure fields of this magnitude. Two magnetometers that were identified as candidates for the space environment are the

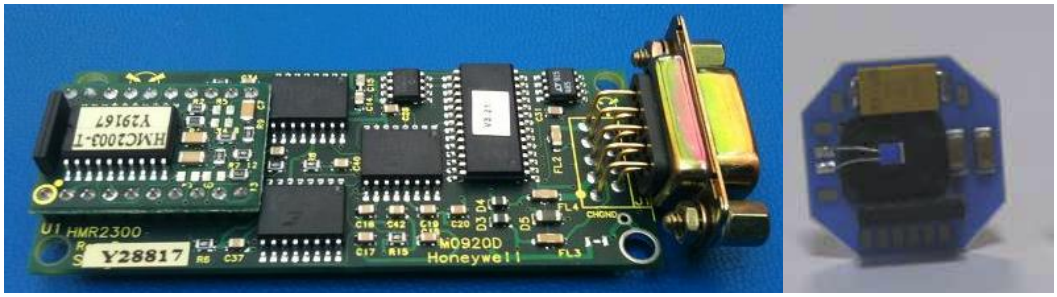


Figure 4.2.1: Honeywell HMR2300 (left) [Photo Credit: Henri Kjellberg] and SS LTD Magnetometer (right)[23].

Performance Index	Honeywell HMR2300	SS LTD Magnetometer
Mass	28 g	15 g
Power Consumption	0.324 W	0.400 W
Sensitivity	6.7 nT	10 nT

Table 4.2.1: Performance characteristics of the magnetometers.[22, 23]

Honeywell HMR 2300 and the Satellite Services LTD Magnetometer.[22, 23] These magnetometers are shown in Figure 4.2.1.

The two sensors have very similar performance characteristics as seen in Table 4.2.1. The error from the magnetometer is driven by two things, the magnetometer performance and the accuracy of the magnetic field model. Both the magnetometer measurement sensitivity and the magnetic field models have an accuracy of approximately 1 degree resulting in an accuracy of less than 5 degrees. Therefore, the magnetometer is a more coarse sensor than the sun sensor, but by combining the magnetometer and sun sensor measurements attitude estimate is improved. The Honeywell HMR 2300 has successful flight heritage on Cute 1.7+APD.[24] From these criteria, the Honeywell magnetometer was chosen, primarily due to flight heritage.

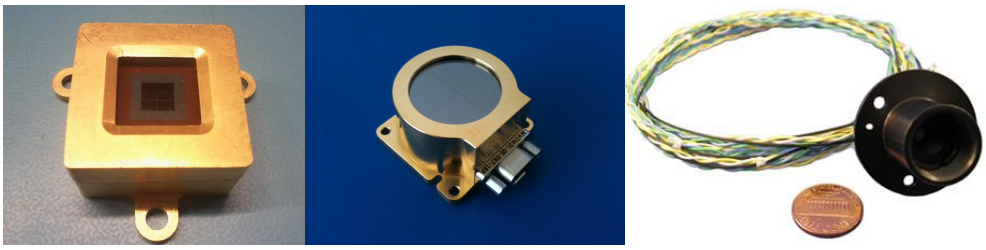


Figure 4.2.2: ISIS (left) [26], SI (middle) [Photo Credit: Henri Kjellberg], Comtech (right) [27] sun sensors.

4.2.1.2 Sun Sensors

The sun sensor produces a unit vector direction measurement from the spacecraft to the centroid of the sun. Given the position of the spacecraft and the position of the sun, attitude knowledge can be obtained from this direction vector. The direction of the sun is known extremely accurately, and therefore these sensors do not suffer from knowledge limitation in the same way that the magnetometer does. Three commercially available sun sensors were identified, the Sinclair Interplanetary (SI) Digital 2-Axis Sun Sensor, the Comtech Aeroastro Medium Sun Sensor, and the Innovative Solutions in Space (ISIS) Miniaturized Analog Fine Sun Sensor.[25, 26, 27] These sun sensors are shown in Figure 4.2.2.

The performance characteristics that are important in a sun sensor include the accuracy of the vector measurement and the size of the field of view. These devices consume little power. In fact the Comtech sun sensor utilizes no power from the spacecraft bus, the light generates power for the device's operation. Of these three devices, the SI sun sensor is the most accurate and has the largest field of view as seen in Table 4.2.2. The SI sun sensor was chosen because it is the most accurate, has the largest field of view. It also produces a digital vector solution,

Performance Index	ISIS	SI	Comtech
Mass	50 g	34 g	36 g
Accuracy	0.3 deg	0.1 deg	2 deg
Field of View	128 deg	140 deg	120 deg

Table 4.2.2: Performance characteristics of the sun sensors.[25, 26, 27]

eliminating the need for the spacecraft to convert an analog measurement to a digital signal.

4.2.1.3 Gyroscopes

In addition to the two vector measurements, the performance of the GN&C module is improved by the ability to take inertial measurements, particularly the body rate of the spacecraft. Typically, attitude determination filters do not estimate the body rate, rather they "fly the gyros." That is, they utilize the gyroscope measurements as the body rate of the spacecraft, with a correction for bias. In cases when two vector measurements are not available to provide an external estimate of the spacecraft attitude, the gyroscopes are utilized to propagate forward the attitude estimate. The more accurate the gyroscope, the less often a new pair of vector measurements is required in order to maintain the necessary accuracy in attitude knowledge. Figure 4.2.3 shows two Analog Devices gyroscopes that were considered for the ADC module, the ADIS16265 and ADIS16130. [28, 29]

Table 4.2.3 summarizes the gyroscope's performance characteristics. Overall, the ADIS16130 is more accurate, but it also has an increased power consumption and a much larger volume. The ADIS16265 is approximately 11 mm x 11 mm whereas the ADIS16130 is approximately 36 mm x 40 mm. The most important

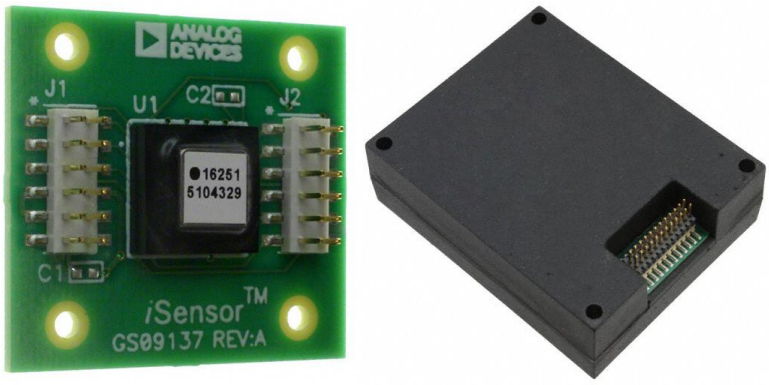


Figure 4.2.3: Analog devices gyroscopes ADIS16265 (left) and ADIS16130 (right).[28, 29]

Performance Index	ADIS16130	ADIS16265
Rate Noise Density	0.0125 deg/sec/$\sqrt{\text{Hz}}$	0.044 deg/sec/$\sqrt{\text{Hz}}$
Angular Random Walk	0.56 deg/$\sqrt{\text{hour}}$	2 deg/$\sqrt{\text{hour}}$
In-Run Bias Stability	0.0016 deg/sec	0.007 deg/sec
Dynamic Range	250 deg/sec	80 deg/sec
Power Consumption	0.365 W	0.205 W
Sensitivity	0.000042 deg/sec/LSB	0.0183 deg/sec/LSB

Table 4.2.3: Important performance characteristics metrics for the analog devices gyroscopes.[28, 29]

performance characteristics are the in-run bias stability and the angular random walk. These determine the statistical accuracy properties of gyroscope measurements in between external updates.

Figure 4.2.4 shows the attitude knowledge error of the two sensors for a 2000 second simulation in which the spacecraft passes into the Earth's shadow at approximately 450 seconds. Note, the performance shown here is only a snapshot and cannot be used to compare with the statistical performance characteristics given in Table 4.1.1 because information is added from the magnetometers, and the

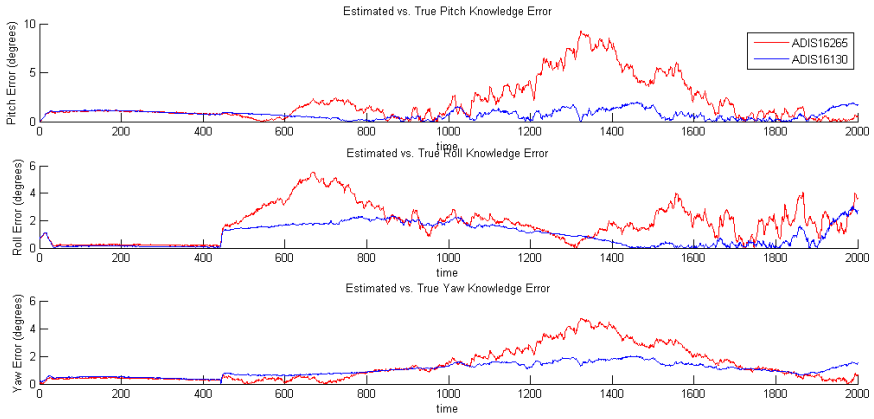


Figure 4.2.4: StarBox simulation showing the attitude knowledge error after loss of line of sight to the sun. Attitude is determined by the magnetometer and gyroscopes only.

system has 3 degrees of freedom in rotation, therefore the errors in each axis are correlated to each other.

Figure 4.2.4 can be used to compare the overall performance characteristics of the two gyroscopes. As is expected the larger, more accurate ADIS16130 gyroscope holds the attitude knowledge closer to the truth than the smaller ADIS16265. The design decision between these two sensors for the first iteration of the ADC module was to go with the smaller, lower power ADIS16265. The performance is good enough to contribute to the operations during periods of sun visibility and maintain rough pointing during eclipse. The performance of the ADIS16130 is definitely better, but the performance increase is not needed to satisfy the requirements of the baseline mission.

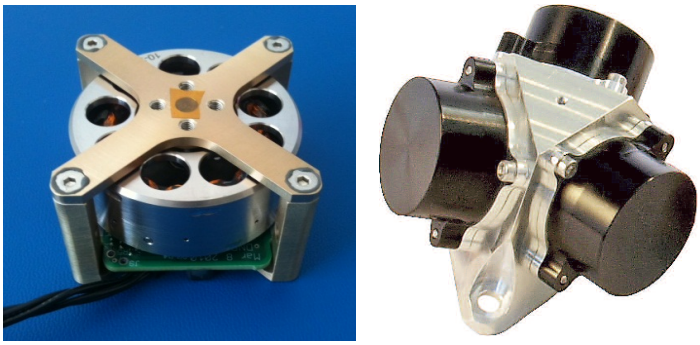


Figure 4.2.5: The SI 10 mNm-s Reaction Wheels (left) [Photo Credit: Henri Kjellberg] and the Astrofein RW1 (right) [30].

4.2.2 Actuators

Generally, small spacecraft are controlled by a system consisting of reaction wheels and magnetorquers. The reaction wheels actively control the spacecraft's attitude by adjusting the total spacecraft angular momentum. As a reaction wheel spins in one direction, the spacecraft body spins in the opposite direction due to the conservation of angular momentum. However, the reaction wheels cannot impart external forces. Eventually, the momentum accrued in the reaction wheels from external disturbances must be dissipated using the external control torques provided by the magnetorquers.

4.2.2.1 Reaction Wheels

Few commercial reaction wheels exist within the size and power constraints of the CubeSat form factor. Two that were identified in this study are the Astro- und Feinwerktechnik Aldershof (Astrofein) RW1 and the Sinclair Interplanetary (SI) 10 mNm-s Reaction Wheel. Both are shown in Figure 4.2.5.[30, 31, 32]

Reaction wheels have three main performance metrics: the maximum torque,

Performance Index	Astrofein	SI
Wheel Max Torque	0.023 mNm	1 mNm
Wheel Nominal Momentum	0.58 mNm-s	10 mNm-s
Set Mass	125 g	360 g
Set Power	1.2 W	0.418 W
Wheel Configuration	4 wheel tetrahedron	3 wheel orthogonal

Table 4.2.4: Performance characteristics of the reaction wheels.[31, 30]

the maximum wheel speed, and the torque noise. The maximum torque determines the time it takes for the wheel speed to accelerate to the desired value. The total momentum drives the fastest rate at which the body can rotate, as well as the ability to absorb the effects of disturbance torques. The torque noise determines the accuracy of the slew and absolute pointing capabilities. The first two performance characteristics as well as the typical mass and power consumption values are summarized in Table 4.2.2. Note that the SI 10 mNm-s reaction wheel is quite a bit larger, and provides a higher torque and momentum storage capability, but has a significantly lower power consumption for a set. The reason behind this is that the tiny reaction wheels from Astrofein, which were designed for a 1-U CubeSat, require significantly higher rotation rates to produce the necessary momentum for a 3-U CubeSat.

The torque and momentum parameters affect the spacecraft's control capabilities in two ways. First, the ability to slew quickly requires both the ability to torque quickly, but also a high wheel speed saturation value. Figure 4.2.6 shows the rotation rate of a 3-U CubeSat during a 180 degree maneuver in the minimum time that the reaction wheel is capable of achieving. The SI reaction wheels have both a higher torque allowing them to reach their maximum rotation rate more quickly

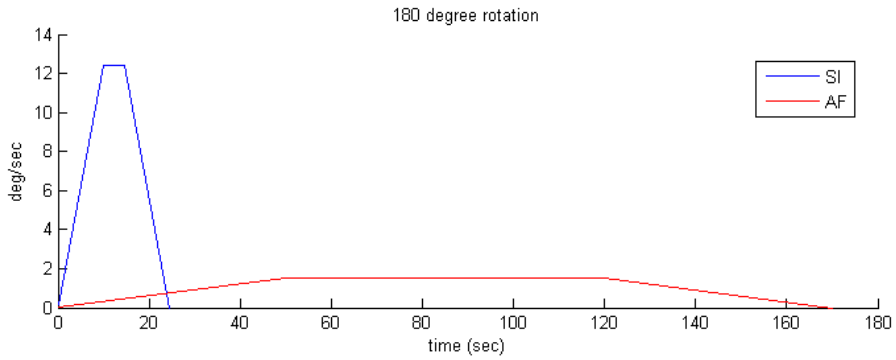


Figure 4.2.6: Body rate during 180 degree rest-to-rest maneuver for a 3-U CubeSat.

and a higher saturation momentum allowing the body rate to be higher. The 180 degree turn takes approximately 25 seconds with the SI wheels whereas with the AF wheels the same maneuver takes approximately 170 seconds.

The second way in which the total momentum affects the wheels capabilities is a result of the total amount of external disturbance torques the spacecraft can store. In the worst-case analysis, for a spacecraft pointing in a Earth referenced direction, the momentum stored in the reaction wheels increases as seen in Figure 4.2.7. These models are conservative, in that the spacecraft parameters assume no attempt to mitigate the residual magnetic dipole, and the center of mass offset is the maximum allowed in the CubeSat specifications.

While it may be possible to achieve good results with the AF reaction wheel set, it would be more risky without fully understanding the performance characteristics in space. The SI reaction wheels allow for the design to be more robust and aggressive at the cost of volume.

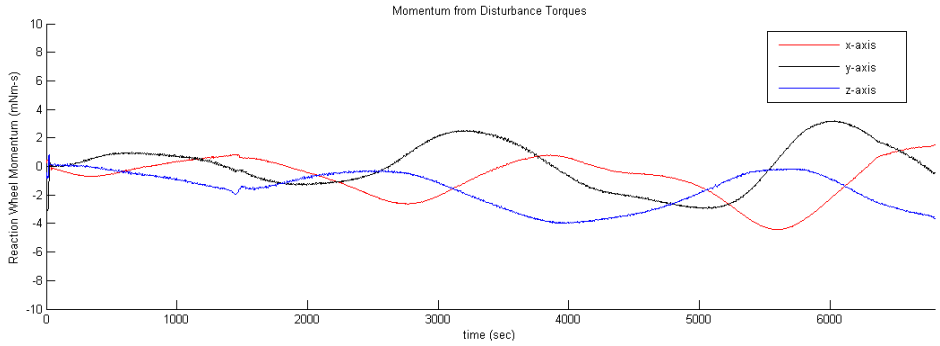


Figure 4.2.7: Worst-case disturbance torques causing momentum in reaction wheels to accrue.

4.2.2.2 Magnetic Torque Rods

Several commercial vendors of magnetorquers exist. However, due to sizing constraints, it is simpler to design a custom magnetorquer to maximize the volume that is available. Therefore, the magnetorquers are not evaluated in this trade study. It is assumed that the final design of the GN&C module will contain custom made magnetorquers.

4.2.3 Embedded Computers

The embedded flight computer for the GN&C module must be able to retrieve all the sensor telemetry, calculate the state estimate, calculate the desired control, and command the actuators. For this design, only industrial hardware was considered. Radiation tolerance was not deemed a priority due to the relatively minor radiation conditions in the 400 km circular reference orbit. Embedded computers are sold in various forms. Here the focus is on "Systems on a Module" (SOM). A SOM is an embedded computer that has all of the necessary integrated circuits on a printed circuit board (PCB) with the processor. The SOM is then at-

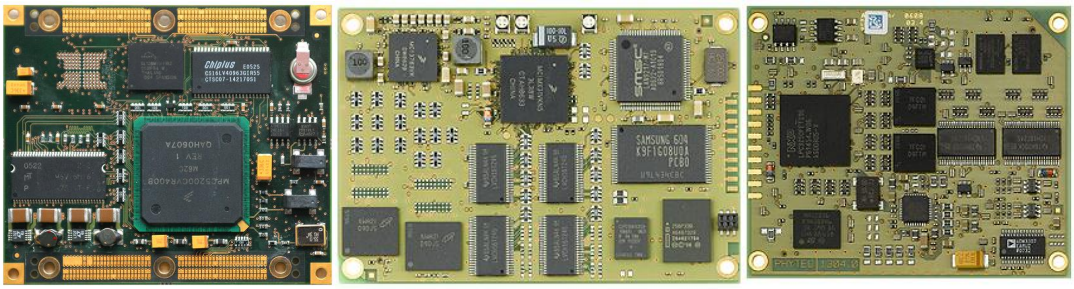


Figure 4.2.8: The MPX5200 (left), the i.MX31 (middle), and the LPC3250 (right).[35, 33, 34]

tached to an interface board that exposes the electrical connections of the SOM to the external peripherals. In this case, the peripherals are the sensors and actuators. Three SOMs were identified and purchased for evaluation, these were the Microsys MPX5200, the phyCORE i.MX31, and the phyCORE LPC3250.[34, 35, 33] The SOMs are shown in 4.2.8.

Table 4.2.5 shows some performance criteria for the SOMs. The performance criteria of an embedded computer are a little more difficult to enumerate than for the sensors and actuators. The processor needs to be fast enough for real time execution of the necessary tasks and have enough available communication ports for all the peripherals. However, these metrics are difficult to quantify because the efficiency of the code can always be improved and the necessary communication infrastructure can be achieved in a myriad of ways regardless of how many peripheral ports the SOM itself has exposed.

Indeed, for this application the most important performance criteria are the power consumption and the ease of developing software on the platform. In particular, interaction with the operating system and access to the embedded computer are

Performance Index	MPX5200	i.MX 31	LPC3250
Clock Speed	400	532	208
USB	2	1	1
UARTS	2	5	4
I2C	2	3	2
SPI	1	3	2
Power	2 W	1.65 W	0.372 W

Table 4.2.5: Performance characteristics of the embedded computers.[35, 33, 34]

extremely important. From these criteria the LPC3250 was chosen because of its lower power consumption, well supported Linux kernel development environment known as the "Linux Target Image Builder" (LTIB), and direct support for using a Network File System (NFS). LTIB allows for a graphical interface into selecting the particular features that are desired for the embedded Linux kernel. NFS allows for rapid code iteration on the embedded system itself. These features are more important than the actual processing performance characteristics of the embedded computer.

Chapter 5

Standard Attitude Determination and Control Algorithms

5.1 Attitude Determination Algorithms

As a baseline, two common attitude determination algorithms have been implemented and tested within the StarBox simulation. These algorithms are the Quaternion Estimation (QUEST) and the Multiplicative Extended Kalman Filter (MEKF). QUEST is used to obtain a snapshot estimate of the attitude using only the most recent vector measurements without using the gyroscope measurements. The MEKF is used to filter out noise using knowledge of the dynamics, the gyroscopes, and weighted vector measurements. The algorithms are not mutually exclusive and in fact they are used together in this design. This chapter first presents the fundamental concept underpinning the attitude determination problem and then presents a pair of algorithms for estimating the attitude of the spacecraft.

5.2 Wahba's Problem

Estimating a spacecraft attitude from a set of vector measurements is known as "Wahba's Problem."[36] With at least two linearly independent vectors, full knowledge of the spacecraft's attitude can be obtained. Understanding the mathematics of Wahba's problem helps in understanding the methods used to estimate

spacecraft attitude on an embedded flight computer. Although there are numerous sources discussing Wahba's problem [37, 38, 39, 40], it is eloquently derived and summarized by Christian [16]. Given a set of vector measurements, the performance index J is minimized as a function of the vehicle's attitude as follows:

$$\min\ J\left(\mathbf{T}_B^I\right) = \frac{1}{2}\sum_{i=1}^{m} w_i \left\| \left(\tilde{\mathbf{e}}_i\right)_B - \mathbf{T}_B^I \left(\mathbf{e}_i\right)_I \right\|^2 . \tag{5.2.1}$$

Here, the unit vector $\left(\tilde{\mathbf{e}}_i\right)_B$ is the ith sensor vector direction measurement to a reference object in the body frame and $\left(\mathbf{e}_i\right)_I$ is the unit vector to the same object obtained from an on-board model of the true direction of the object in the inertial frame. w_i is the weight given to the ith measurement. Wahba's problem attempts to find a rotation matrix,$\mathbf{T}_B^I$, that best represents the attitude of the spacecraft given the imperfect measurements and their weights. Through a series of manipulations that are beyond the scope of this thesis, 5.2.1 can be represented in the form[16]

$$\min\ J\left(\mathbf{T}_B^I\right) = \lambda_o - \mathrm{tr}\left[\mathbf{T}_B^I \mathbf{B}^T\right]. \tag{5.2.2}$$

Here, the trace operator $\mathrm{tr}\left[\mathbf{A}\right]$ is the sum of the diagonal elements of $\mathbf{A}$. A new matrix is defined known as the attitude profile matrix, which contains the information from the sensor measurements and the on-board models:

$$\mathbf{B} = \sum_{i=1}^{m} w_i \left(\tilde{\mathbf{e}}_i\right)_B \left(\mathbf{e}_i\right)_I^T . \tag{5.2.3}$$

Representing the problem in terms of quaternions allows more insight in how to find an attitude that minimizes the performance index. Recall that a rotation

matrix can be represented in terms of a quaternion as [16]

$$T(\bar{\mathbf{q}}) = \left(q_4^2 - \mathbf{q}^T\mathbf{q}\right)\mathbf{I}_{3x3} + 2\mathbf{q}\mathbf{q}^T + 2q_4\left[\mathbf{q}\times\right]. \tag{5.2.4}$$

Using this representation in and substituting into 5.2.2 after some additional manipulation one gets the performance index J as a function of the attitude quaternion $\bar{\mathbf{q}}$:

$$J(\bar{\mathbf{q}}) = \lambda_o - \bar{\mathbf{q}}^T\mathbf{K}\bar{\mathbf{q}}. \tag{5.2.5}$$

$\mathbf{K}$ is known as the 4×4 Davenport matrix:

$$\mathbf{K} = \begin{bmatrix} \mathbf{S} - \mu\mathbf{I}_{3\times 3} & \mathbf{z} \\ \mathbf{z}^T & \mu \end{bmatrix}. \tag{5.2.6}$$

The components of the Davenport matrix are defined as:

$$\mathbf{S} = \mathbf{B} + \mathbf{B}^T, \tag{5.2.7}$$

$$\mu = \mathrm{tr}\left[\mathbf{B}\right], \tag{5.2.8}$$

$$\left[\mathbf{z}\times\right] = \mathbf{B} - \mathbf{B}^T. \tag{5.2.9}$$

The unity norm constraint of the quaternion must be explicitly adjoined to the performance index using a Lagrange multiplier (λ):

$$J(\bar{\mathbf{q}},\lambda) = \lambda_o - \bar{\mathbf{q}}^T\mathbf{K}\bar{\mathbf{q}} + \lambda\left(\bar{\mathbf{q}}^T\bar{\mathbf{q}} - 1\right). \tag{5.2.10}$$

By taking the first differential a set of three equations are obtained:

$$(\mathbf{S}-\mu\mathbf{I}_{3\times 3})\,\mathbf{q}+z q_4 = \lambda\mathbf{q}, \tag{5.2.11}$$

$$\mathbf{z}^T\mathbf{q}+\mu q_4 = \lambda q_4, \tag{5.2.12}$$

and the unity norm constraint

$$\mathbf{q}^T\mathbf{q}+q_4^2 = 1. \tag{5.2.13}$$

By combining 5.2.11 and 5.2.12, a simple eigenvalue problem emerges

$$\mathbf{K}\bar{\mathbf{q}} = \lambda\bar{\mathbf{q}}. \tag{5.2.14}$$

Substituting the 5.2.14 into 5.2.5 the simple performance index becomes

$$J(\bar{\mathbf{q}}) = \lambda_o - \lambda. \tag{5.2.15}$$

Merely by inspection, one can note that the solution to this optimization problem is the largest eigenvalue of the Davenport matrix. Indeed, the quaternion estimation algorithms are merely different methods for finding the largest eigenvalue of the Davenport matrix.

5.3 QUaternion ESTimation (QUEST)

The QUEST algorithm is a method for finding the largest eigenvalue of the Davenport matrix efficiently.[39] QUEST does not utilize knowledge of the dynamics or measurements from the gyroscope. Instead it provides a weighted best

estimate of the spacecraft's attitude quaternion using measurements with the same time stamp. QUEST is a simple algorithm in which very little can go wrong. There are cases in which a singularity appears. The singularity occurs when there is a 180 degree rotation appears in the attitude solution. This difficulty can be overcome by in these cases using an alternative algorithm, or by performing a rotation. Nevertheless, QUEST provides a good check for more complicated Kalman filter methods to make certain the estimated attitude solution on-board is not corrupt. Furthermore, the quaternion solutions from QUEST can be fed into a Kalman filter in order increase the fidelity of the attitude estimate. The QUEST algorithm was implemented from another excellent derivation by Christian [41] which is summarized as follows:

Step 1: From the sensor measurements and the reference models compute **B** and **z**:

$$\mathbf{B} = \sum_{i=1}^{n} w_i \left(\tilde{\mathbf{e}}_i\right)_{B,i} \left(\mathbf{e}_i\right)_I^T, \tag{5.3.1}$$

$$\mathbf{z} = \sum_{i=1}^{n} w_i \left(\left(\tilde{\mathbf{e}}_i\right)_{B,i} \times \left(\mathbf{e}_i\right)_I\right). \tag{5.3.2}$$

Step 2: Obtain additional constants related directly to the attitude profile matrix:

$$\mathbf{S} = \mathbf{B} + \mathbf{B}^T, \tag{5.3.3}$$

$$\mu = \mathrm{tr}\left[\mathbf{B}\right], \tag{5.3.4}$$

$$\delta = \det\left[\mathbf{S}\right], \tag{5.3.5}$$

$$k = -\frac{1}{2}\left[\mathrm{tr}\left[\mathbf{S}^2\right] - \mathrm{tr}\left[\mathbf{S}\right]^2\right]. \tag{5.3.6}$$

Step 3: Compute the constants required for the Newton-Raphson iteration:

$$\begin{aligned} c_2 &= -2\mu^2 + k - z^T z, && (5.3.7)\\ c_1 &= -\delta + \mathbf{z}^T \mathbf{S}\mathbf{z}, && (5.3.8)\\ c_0 &= -\mu\left(k\mu - \delta - \mu^3\right) - \mathbf{z}^T\left[\left(k-\mu^2\right)\mathbf{I}_{3\times 3} - \mu\mathbf{S} + \mathbf{S}^2\right]\mathbf{z}. && (5.3.9) \end{aligned}$$

Step 4: Perform Newton-Raphson iteration until $\|\lambda_{k+1} - \lambda_k\| < \text{tol}$,

$$\lambda_{k+1} = \lambda_k - \frac{\lambda_k^4 + c_2\lambda_k^2 + c_1\lambda_k + c_0}{4\lambda_k^3 + 2c_2\lambda_k + c_1}. \tag{5.3.10}$$

Step 5: From the largest converged eigenvalue, λ, compute additional constants:

$$\begin{aligned} \alpha &= \lambda^2 - \mu^2 + k,\\ \beta &= \lambda - \mu,\\ \gamma &= (\lambda+\mu)\left(\lambda^2 - \mu^2 + k\right) - \delta, && (5.3.11)\\ \mathbf{y} &= \left[\alpha\mathbf{I}_{3\times 3} + \beta\mathbf{S} + \mathbf{S}^2\right]\mathbf{z}. \end{aligned}$$

Step 6: Solve for the optimal quaternion using the weighted measurements,

$$\bar{\mathbf{q}} = \frac{1}{\sqrt{\gamma^2 + \mathbf{y}^T\mathbf{y}}}\begin{bmatrix}\mathbf{y}\\ \gamma\end{bmatrix}. \tag{5.3.12}$$

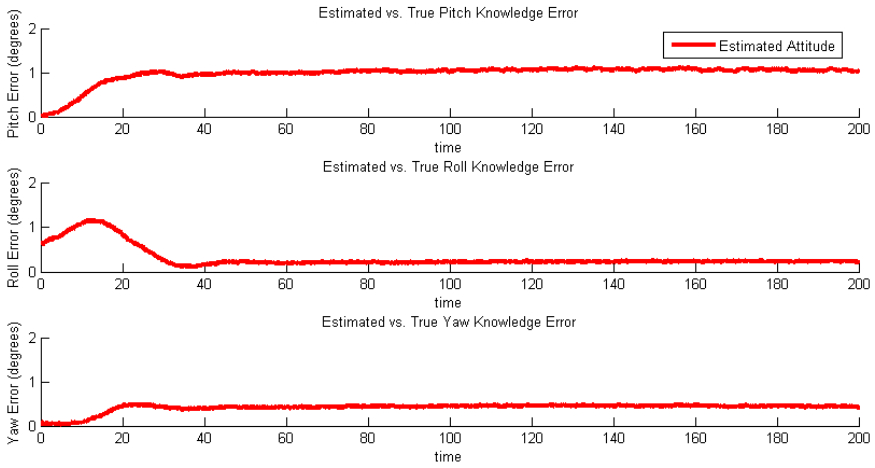


Figure 5.4.1: Attitude estimate using vector MEKF in StarBox with full sensor noise and bias settings during a maneuver.

5.4 Multiplicative Extended Kalman Filter (MEKF)

A MEKF allows for knowledge of rotational dynamics and gyroscope measurements to be incorporated into the spacecraft attitude estimate. Doing so allows for a reduction in the noise in the estimate because more measurements are included as well as time correlations in the vehicle's equations of motion. An example of the filter functioning through an attitude maneuver with full noise and bias settings in StarBox is shown in Figure 5.4.1. Here, a slew maneuver is performed that lasts approximately 40 seconds followed by steady state pointing. A nonzero pointing error persists because of the bias errors in the magnetometer measurements, magnetic field model, and sun sensor measurements.

There are two baseline methods used in the MEKF implemented in StarBox. The first is the quaternion MEKF and the second is the vector MEKF. The difference between the two is the form of measurement input into the filter. With

the quaternion MEKF, the filter is fed quaternion estimates from the sensors via QUEST. In the vector form, the measurement vectors are fed directly into the filter. The advantage with the vector filter is that individual measurements can be fed into the filter, whereas with the quaternion MEKF one needs at least 2 vector measurements before proceeding. Therefore, magnetometer measurements can still be utilized in the filter when the spacecraft does not have visibility of the sun for the sun sensor measurements. The vector MEKF was derived from the quaternion MEKF. The differences of the two algorithms are subtle, but they are highlighted in the procedure provided below.[41]

Step 1: From the measurement, produce the attitude representation, the sensitivity matrix, and Kalman gain:

- In the case of a quaternion measurement, $\bar{\mathbf{q}}_{obs,k}$, with a covariance $\mathbf{R}_k$ obtained from QUEST, and an a priori quaternion estimate $\hat{\bar{\mathbf{q}}}_k^-$, the attitude representation, $\mathbf{y}_k$, is three-dimensional and the sensitivity matrix are computed as follows:

$$\delta\bar{\mathbf{q}}_{obs,k} = \bar{\mathbf{q}}_{obs,k} \otimes \left(\hat{\bar{\mathbf{q}}}_k^-\right)^{-1}, \tag{5.4.1}$$

$$\mathbf{y}_{k_q} = 2\left(\frac{\delta\mathbf{q}_{obs,k}}{\left\|\delta\mathbf{q}_{obs,k}\right\|}\right)\arctan\left[\frac{\left\|\delta\mathbf{q}_{obs,k}\right\|}{\delta q_{obs,k}}\right], \tag{5.4.2}$$

$$\mathbf{H}_{k_q} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}. \tag{5.4.3}$$

- In the case of a vector measurement from the sun sensor, $\tilde{\mathbf{e}}_{ss,k}$, and magnetometer $\tilde{\mathbf{e}}_{mag,k}$, the attitude representation, $\mathbf{y}_k$, is $3\cdot m$-dimensional, where m is the number of measurements taken at a single time step. For the GN&C

module implemented in StarBox, $\mathbf{y}_k$ is a 6 dimensional vector. The attitude representation and the sensitivity matrix are computed as follows:

$$\mathbf{y}_{k_v} = \begin{bmatrix} \tilde{\mathbf{e}}_{ss,k} \\ \tilde{\mathbf{e}}_{mag,k} \end{bmatrix}, \tag{5.4.4}$$

$$\mathbf{h}_k = \begin{bmatrix} \mathbf{T}\left(\hat{\bar{\mathbf{q}}}_k^-\right)\left(\mathbf{e}_{ss,k}\right)_I \\ \mathbf{T}\left(\hat{\bar{\mathbf{q}}}_k^-\right)\left(\mathbf{e}_{mag,k}\right)_I \end{bmatrix}_{6\times 1}, \tag{5.4.5}$$

$$\mathbf{H}_{k_v} = \begin{bmatrix} \begin{bmatrix} \mathbf{h}_{k_{(1:3)}} \times \end{bmatrix} & 0_{3\times 3} \\ \begin{bmatrix} \mathbf{h}_{k_{(4:6)}} \times \end{bmatrix} & 0_{3\times 3} \end{bmatrix}_{6\times 6}. \tag{5.4.6}$$

The covariance must be obtained from the angular variance of the measurement error in the sun sensor and magnetometer,

$$\mathbf{R}_k = \begin{bmatrix} \sigma_{ss,k}^2 \left[\mathbf{I}_{3\times 3} - \left(\mathbf{e}_{ss,k}\right)_B \left(\mathbf{e}_{ss,k}\right)_B^T\right] & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \sigma_{mag,k}^2 \left[\mathbf{I}_{3\times 3} - \left(\mathbf{e}_{mag,k}\right)_B \left(\mathbf{e}_{mag,k}\right)_B^T\right] \end{bmatrix}. \tag{5.4.7}$$

Note, the vector MEKF can be utilized even when only one vector measurement is available, for example when the sun sensor cannot produce a measurement during eclipse. In that case the components relating to the sun sensor are ignored in the equations above. The quaternion MEKF cannot be used when only one vector measurement is available because no quaternion estimate can be produced using QUEST.

Step 2: Compute the Kalman gain in the standard form,

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k\right]^{-1}. \tag{5.4.8}$$

Step 3: Compute the update to the state vector.

- In the case of a quaternion measurement, the state update is given by

$$\delta\mathbf{x} = \begin{bmatrix} \delta\hat{\boldsymbol{\theta}}_k \\ \delta\hat{\boldsymbol{\beta}}_k \end{bmatrix} = \mathbf{K}\left(\mathbf{y}_k\right). \tag{5.4.9}$$

Here, $\delta\hat{\boldsymbol{\theta}}_k$ is an update to the attitude representation vector, and $\delta\hat{\boldsymbol{\beta}}$ is an update to the gyroscope bias estimate.

- In the case of a vector measurement, the state update is given by

$$\delta\mathbf{x} = \begin{bmatrix} \delta\hat{\boldsymbol{\theta}}_k \\ \delta\hat{\boldsymbol{\beta}}_k \end{bmatrix} = \mathbf{K}\left(\mathbf{y}_k - \mathbf{h}_k\right). \tag{5.4.10}$$

Beyond this point, the process is similar for both forms of the MEKF.

Step 4: Update the bias and quaternion estimate.

$$\hat{\boldsymbol{\beta}}_k^+ = \hat{\boldsymbol{\beta}}_k^- + \delta\hat{\boldsymbol{\beta}}_k \tag{5.4.11}$$

$$\hat{\bar{\mathbf{q}}}_k^+ = \hat{\bar{\mathbf{q}}}_k^- + \frac{1}{2}\boldsymbol{\Xi}\left(\hat{\bar{\mathbf{q}}}_k^-\right)\delta\hat{\boldsymbol{\theta}}_k \tag{5.4.12}$$

$$\hat{\bar{\mathbf{q}}}_k^+ = \frac{\hat{\bar{\mathbf{q}}}_k^+}{\left\|\hat{\bar{\mathbf{q}}}_k^+\right\|} \tag{5.4.13}$$

Here,

$$\boldsymbol{\Xi}\left(\hat{\bar{\mathbf{q}}}_k^-\right) = \begin{bmatrix} \hat{q}_{k,4}^- & -\hat{q}_{k,3}^- & \hat{q}_{k,2}^- \\ \hat{q}_{k,3}^- & \hat{q}_{k,4}^- & -\hat{q}_{k,1}^- \\ -\hat{q}_{k,2}^- & \hat{q}_{k,1}^- & \hat{q}_{k,4}^- \\ -\hat{q}_{k,1}^- & -\hat{q}_{k,2}^- & -\hat{q}_{k,3}^- \end{bmatrix} \tag{5.4.14}$$

Note, the quaternion must be normalized in Equation 5.4.13 which is generally not recommended. Mathematically, information is being lost during the normalization process. This is one reason the Sequential Optimal Attitude Routine

(SOAR) algorithm was developed by Christian and Lightsey.[16] However, SOAR is beyond the scope of this thesis.

Step 5: Update the covariance using the standard form,

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_k^-\,(I - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T. \tag{5.4.15}$$

Step 6: Propagate the estimate forward by first calculating the derivatives of the quaternion and covariance and then utilizing a numerical integrator. In this step, the measurements from the gyroscopes are directly incorporated into filter. The gyroscope measurements are corrected for bias using the bias estimate from Step 4.

$$\hat{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}_{obs}(t) - \hat{\boldsymbol{\beta}}(t) \tag{5.4.16}$$

Note, that time dependence in $\boldsymbol{\omega}_{obs}(t)$ indicates that the gyroscope measurements are sampled at a higher rate throughout the integration process. Each time step in the Runge-Kutta integrator can use the latest gyroscope measurement. The derivative of the quaternion is given by,

$$\dot{\hat{\mathbf{q}}}(t) = \frac{1}{2}\boldsymbol{\Xi}\left(\hat{\mathbf{q}}_k^-\right)\hat{\boldsymbol{\omega}}(t). \tag{5.4.17}$$

And the derivative of the covariance is given by,

$$\dot{\mathbf{P}}(t) = \mathbf{F}(\hat{x},t)\,\mathbf{P}(t) - \mathbf{P}(t)\,\mathbf{F}(\hat{x},t)^T + \mathbf{G}(t)\,\mathbf{Q}(t)\,\mathbf{G}(t)^T \tag{5.4.18}$$

where,

$$\mathbf{F}(\hat{x},t) = \begin{bmatrix} [\hat{\boldsymbol{\omega}}\times] & -\mathbf{I}_{3\times3} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, \tag{5.4.19}$$

$$\mathbf{G}(t) = \begin{bmatrix} -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}, \tag{5.4.20}$$

$$\mathbf{Q}(t) = \begin{bmatrix} \sigma_{\omega}^2\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \sigma_{\hat{\beta}}^2\mathbf{I}_{3\times3} \end{bmatrix}. \tag{5.4.21}$$

Providing estimates for the values in the **Q** matrix is difficult because the statistical performance of the gyroscope will vary depending on environmental factors such as temperature. The MEKF is an excellent baseline navigation filter for the GN&C module. The quaternion attitude representation prevents singularity conditions and simplifies the flight software. In the future, more advanced navigation filters such as SOAR will be explored for implementation.

5.5 Basic Attitude Control Algorithms

Attitude control algorithms utilize the best estimate of the state of the spacecraft and knowledge about its mechanical properties in order to compute actuator torques to drive the current attitude to a desired attitude. In this chapter, a basic quaternion attitude controller and a slightly more complex torque and body rate constrained quaternion attitude controller are presented.

5.5.1 Basic Quaternion Attitude Controller

Attitude control is performed using quaternion feedback control because the quaternion representation does not suffer from singularities the way Euler angles do. A simple quaternion feedback control assumes a rigid spacecraft with dynamics described by Euler's equation of rotational motion,

$$\mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} = \mathbf{u}. \tag{5.5.1}$$

Here $\mathbf{u}$ is the control torques applied by the reaction wheels. A simple linear feedback controller described by Wie [42] has the form,

$$\mathbf{u} = -\mathbf{K}\mathbf{q}_e - \mathbf{C}\boldsymbol{\omega}. \tag{5.5.2}$$

$\bar{\mathbf{q}}_e$ is the quaternion error, that is the quaternion that describes the difference between the desired quaternion, $\bar{\mathbf{q}}_d$ and the current quaternion, $\bar{\mathbf{q}}_c$,

$$\mathbf{q}_e = \begin{bmatrix} q_{4c} & q_{3c} & -q_{2c} & -q_{1c} \\ -q_{3c} & q_{4c} & q_{1c} & -q_{2c} \\ q_{2c} & -q_{1c} & q_{4c} & -q_{3c} \\ q_{1c} & q_{2c} & q_{3c} & q_{4c} \end{bmatrix} \begin{bmatrix} q_{1d} \\ q_{2d} \\ q_{3d} \\ q_{4d} \end{bmatrix} \tag{5.5.3}$$

The gain matrices **K** and **C** are chosen to alter the behavior of the controller. Note that the above controller does not take into account the angular momentum stored inside the spacecraft, nor does it constrain the output torque or slew rate of the spacecraft. However, it can be shown that it can always be made globally asymptotically stable by selection of proper **K** and **C.**

5.5.2 Constrained Control and Slew Rate Quaternion Controller

Spacecraft can have constraints on their dynamics as a result of sensor and actuator limitations. The basic quaternion attitude controller can be modified in order to maintain the torque commands below a threshold value and to constrain the eigenaxis rotation rate. These two common constraints are the result of the maximum torque capability of an actuator such as a reaction wheel and the maximum angular rate measurement capabilities of a gyroscope. It is important to note that the body rate constraint produced by this controller does not actually limit the rotation rate in any given axis individually. Instead, it limits the aggregate rotation rate.

A modified controller can be formed in two parts by first beginning with the basic quaternion attitude controller. Begin by focusing on the need for a body rate constraint. Wie [42] demonstrates that an eigenaxis slew with a body rate constraint takes the form,

$$\mathbf{u}_c = -\mathbf{K}\text{sat}\left(\mathbf{Pq}\right) - \mathbf{C}\boldsymbol{\omega} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}. \tag{5.5.4}$$

Here, the saturation function is defined as,

$$\text{sat}\left(x\right) = \begin{cases} x^+ & x > x^+ \\ x & x^- < x < x^+ \\ x^- & x < x^- \end{cases}. \tag{5.5.5}$$

The gain matrices **P**, **K**, and **C**, and must be chosen in the form:

$$\begin{aligned} \mathbf{P} &= \text{diag}\left(p_1, p_2, p_3\right), \\ \mathbf{K} &= \text{diag}\left(k_1, k_2, k_3\right)\mathbf{J}, \\ \mathbf{C} &= c\mathbf{J}. \end{aligned} \tag{5.5.6}$$

In order to choose the maximum rotation rate, the components of **K** must be,

$$k_i = c\frac{\left|q_i\left(t_o\right)\right|}{\left\|\mathbf{q}\left(t_o\right)\right\|}\dot{\theta}_{max}. \tag{5.5.7}$$

The gain matrices **K** and **P** must satisfy the equation,

$$\mathbf{KP} = k\mathbf{J}. \tag{5.5.8}$$

where k is a positive scalar.

The above controller provides a 3-dimensional vector of torques that are aligned with the principal moments of inertia. However, reaction wheels in a spacecraft are not necessarily orthogonal to each other. Some spacecraft utilize more than three reaction wheels in order to provide for redundancy should a wheel fail. As long as three linearly independent reaction wheels are available, the spacecraft can be controlled in all axes. The total torque given by the n reaction wheels is given by,

$$\mathbf{u} = \mathbf{a}_1\tau_1 + \mathbf{a}_2\tau_2 + \cdots + \mathbf{a}_n\tau_n. \tag{5.5.9}$$

The unit vector $\mathbf{a}_i$ determines the direction of the wheel torque, τ_i, in the body frame for the ith reaction wheel. Then, the torque distribution matrix is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix}. \tag{5.5.10}$$

In order to compute the amount of torque assigned to each reaction wheel, a Moore-Penrose inverse is performed on the torque distribution matrix,

$$\mathbf{A}^+ = \mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T\right)^{-1}. \tag{5.5.11}$$

The commanded torque from the body rate constrained controller is translated into individual reaction wheel torque commands,

$$\boldsymbol{\tau}_c = \mathbf{A}^+\mathbf{u}_c. \tag{5.5.12}$$

In order to constrain the torque produced without causing a change in the pathway in which the slew is being performed, another saturation operation must be performed. The saturation operator is applied to the result from the first part of the controller,

$$\begin{aligned} \mathbf{u} &= \text{sat}_{\sigma}(\mathbf{u}_c) \\ \mathbf{u} &= -\text{sat}_{\sigma}(\mathbf{K}\text{sat}(\mathbf{Pq}) + \mathbf{C}\omega). \end{aligned} \tag{5.5.13}$$

The saturation operator is given by,

$$\text{sat}_{\sigma}(\boldsymbol{\tau}_c) = \begin{cases} \boldsymbol{\tau}_c & \text{if } \sigma(\mathbf{q}, \boldsymbol{\omega}) \leq 1 \\ \boldsymbol{\tau}_c/\sigma & \text{if } \sigma(\mathbf{q}, \boldsymbol{\omega}) > 1 \end{cases} \tag{5.5.14}$$

Here, the saturation acts by evaluating whether or not any of the reaction wheels have been commanded beyond their capabilities. This is determined by the $\sigma(\mathbf{q}, \boldsymbol{\omega})$ function described by

$$\sigma(\mathbf{q}, \boldsymbol{\omega}) = \|\mathbf{T}\boldsymbol{\tau}_c\|_{\infty} \tag{5.5.15}$$

where,

$$\mathbf{T} = \begin{bmatrix} \tau_1^{max} & & & \\ & \tau_2^{max} & & \\ & & \ddots & \\ & & & \tau_n^{max} \end{bmatrix}. \tag{5.5.16}$$

The $\sigma(\mathbf{q}, \boldsymbol{\omega})$ function finds the torque command with the largest magnitude in the reaction wheel set and then checks if it is beyond the limit. If it is beyond

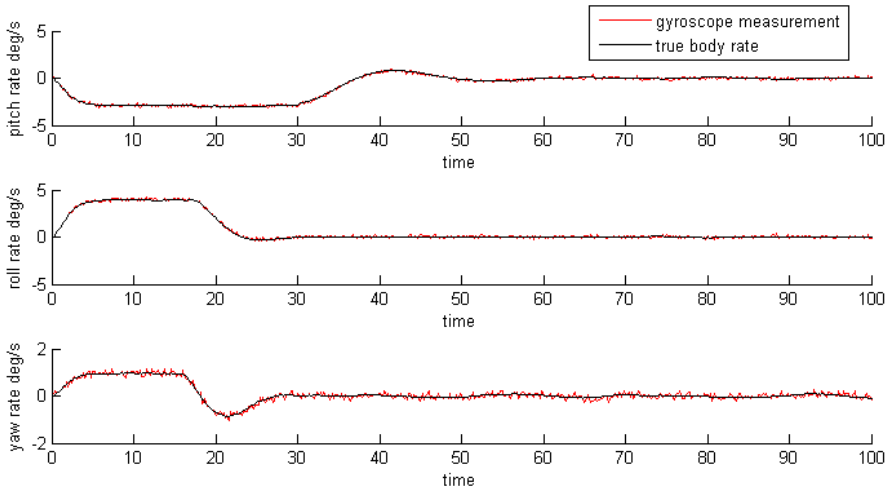


Figure 5.5.1: 65 degree slew with 0 mNm-s initial reaction wheel momentum (left) and with 5 mNm-s initial reaction wheel momentum (right).

the limit, then the saturation command adjusts each reaction wheel appropriately in order to maintain steady motion without deviation from the slew trajectory.

The performance of the body rate constraint can be seen in Figure 5.5.1. In this simulation, the rate was constrained to be less than 5 degrees/second in the eigenaxis rotation rate. Therefore, each individual rate must be less than that as well. The torque constraint can be seen in Figure 5.5.2 as the torque does not exceed 1 mNm in any of the actuators throughout the same slew. Both constraints are satisfied throughout the maneuver simultaneously.

These controllers however do not take into account the change in the dynamics due to the momentum stored in the reaction wheels. Given the relatively small spacecraft size, and the relatively large momentum storage capabilities of the SI reaction wheels, the effects can become noticeable. Figure 5.5.3 shows the deviation

in the slew trajectory for a CubeSat with 3 reaction wheels that have an initial momentum stored. The torque history in Figure 5.5.2 also shows significant change. It is difficult to find a good gain setting that accommodates all the momentum scenarios that the spacecraft can experience. During periods of high momentum storage, the agility of the spacecraft is reduced.

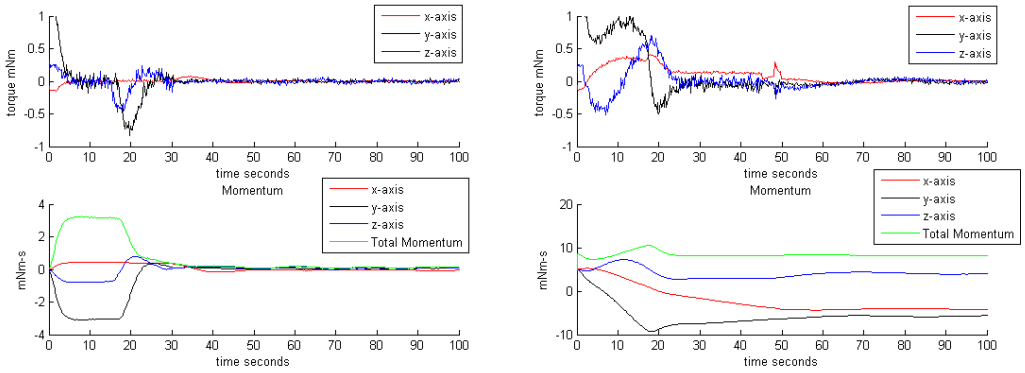


Figure 5.5.2: Torque and momentum history during a 65 degree slew with 0 mNm-s initial momentum (left) and 5 mNm-s initial momentum (right).

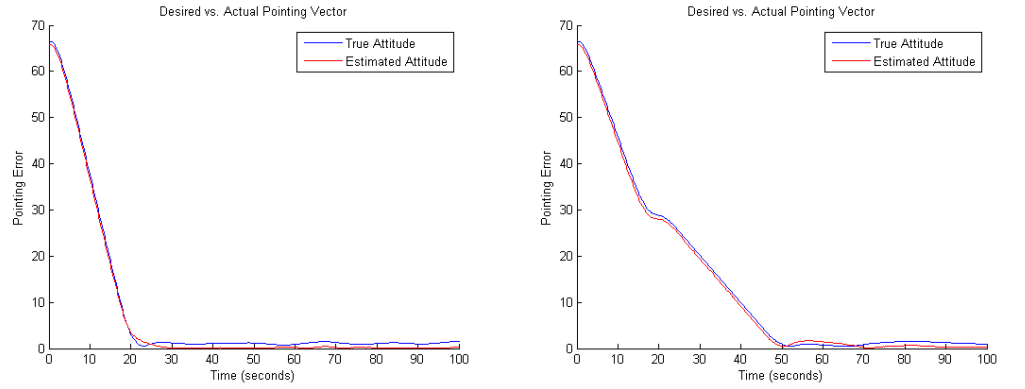


Figure 5.5.3: 65 degree slew with 0 mNm-s initial reaction wheel momentum (left) and with 5 mNm-s initial reaction wheel momentum (right).

Chapter 6

Constrained Attitude Control Algorithms

The constrained controller algorithm presented in Chapter 5 is a good starting point for a spacecraft controller design. However, as spacecraft missions become more capable and require increasing levels of autonomy, the control algorithms must become more advanced. This chapter introduces several constrained attitude control algorithms. Particular focus is placed on method of semi-definite programming. The resulting algorithm is implemented and evaluated on an embedded system in a hardware-in-the-loop simulation.

6.1 Motivation

Spacecraft attitude control constraints are the result of physical limitations imposed on the spacecraft by its mission, sensors, actuators, and payload. For example, there are attitude constraints imposed by scientific instruments. A sensitive optical sensor could become inoperable if exposed to bright celestial objects, such as the Sun, Moon, or Earth. Therefore, whenever the spacecraft must be reoriented, care must be taken to avoid exposing the sensitive sensors to damaging light. Autonomous control algorithms should be able to reorient the spacecraft by taking a feasible attitude trajectory that will not expose the sensitive instrument to a bright source.. Autonomous constrained attitude control is difficult to define; there are

varying degrees of autonomy. Constrained attitude control algorithms allow spacecraft to satisfy the attitude constraints without direct intervention.

Dealing with attitude constraints autonomously allows mission controllers to focus on high level commands to the spacecraft and allows the spacecraft itself to ensure that operational constraints are maintained. Taking care of attitude constraints through software can decrease the hardware cost of a spacecraft mission. SpaceDev's "Trailblazer" microsatellite was able to use hardware that demanded less power, used less volume, and cost less due to their ability to slew the spacecraft while simultaneously satisfying pointing constraints.[43] Costly modifications to their science instrument were avoided by guaranteeing that sun avoidance was achieved through constrained attitude control. Similarly, a spacecraft like Bevo-2 can benefit from the ability to satisfy both control and state constraints autonomously in real-time. Bevo-2 is being designed to fly with only two digital sun sensors and a star tracker without a large baffle in order to decrease cost and mass. The resulting pointing control constraints are handled in software.

Operational modes can be implemented in terms of attitude constraints. Instead of avoiding a certain direction, the spacecraft can be told to maintain pointing inside a specified attitude cone. For example, during the periods of ground communication a spacecraft may need to maintain an antenna pointing within a specified attitude cone. This requirement can be formed as an attitude control constraint. By assembling these requirements together, mission controllers can describe in a very high level the spacecraft requirements and allow the spacecraft itself to find the optimal way to achieve those requirements. Figure 6.1.1 shows some of these common constraints on an arbitrary CubeSat.

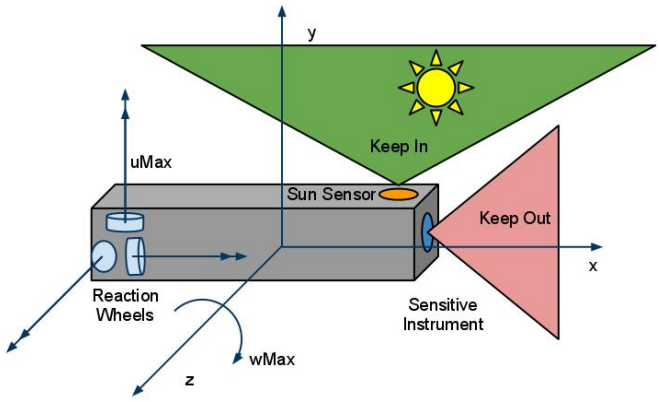


Figure 6.1.1: Actuator performance constraints, spacecraft body rate constraints, and device keep out/in constraints on an arbitrary CubeSat.

There exists sufficient need for an algorithm that is portable, reliable, and computationally realistic. CubeSats in particular are not capable of flying extremely powerful computers due to their electrical power constraints. Therefore, any algorithms must be capable of running on a small processor quickly to meet real-time requirements and leave enough computation margin to allow for other tasks, such as navigation, to run alongside the attitude controller.

The strength of the CubeSat platform is its low cost and high launchability. In the future, large constellations of CubeSats will be deployed to obtain observations which are distributed globally. As the size of the constellation increases, the amount of control from the ground must decrease to maintain operational costs. Satisfying the attitude constraints of individual CubeSats in a constellation cannot practically be performed from the ground in real-time. More of the spacecraft's operations need to become autonomous; only high level strategic commands should be sent from the ground.

6.2 Problem Statement

There are four constraints that can be utilized to describe hardware limitations. The state inequality constraint can be utilized to describe an absolute limit for a particular parameter. The torque produced by a single reaction wheel needs to respect the constraint,

$$|f[u(t)]| < \tau_{max}. \tag{6.2.1}$$

Here, $u(t)$ is the control torque command history and τ_{max} is the maximum reaction wheel torque.

Another state inequality constraint limits the body rate about an axis. The rate must be within the dynamic range of the gyroscope or max slew rate of the vehicle in order to avoid bad navigation measurements. This state inequality constraint is simply described by

$$|g[\omega(t)]| < \dot{\phi}_{max}. \tag{6.2.2}$$

$g[\omega(t)]$ is a function of the spacecraft rotation rate and $\dot{\phi}_{max}$ is the maximum rate set by the user.

The inequality constraint can be integrated in order to form an integral inequality constraint. Integral inequality constraints are perfect for describing the reaction wheel speed saturation limits. Here the torque history is limited by the reaction wheel speed with an equation like,

$$\left| \int_{t_o}^{t} f[u(\tau)]\, d\tau \right| < h_{w_{max}}. \tag{6.2.3}$$

$f[u(t)]$ is a function of the control torques in the reaction wheel and $h_{w_{max}}$ is the maximum momentum that can be stored in the reaction wheel.

Spacecraft state constraints can include properties that are related to the attitude of the spacecraft. One example is hard pointing exclusion zones. These constraints are used to describe sensors that must not be exposed anywhere within a certain angle of a target celestial object. This could be the result of a sensitive instrument or spacecraft keep-out regions during proximity operations. Hard attitude constraints can be expressed as

$$\mathbf{v}(t)^T \mathbf{w}(t) \leq \cos\theta. \tag{6.2.4}$$

Here, $\mathbf{v}(t)$represents the boresight vector of the sensitive instrument in the body frame, $\mathbf{w}(t)$ represents the inertial vector to the bright object to be avoided, and θ is the minimum angle of separation required from the bright object. A weak, or "timed" version of this constraint can be formed as

$$\int_{t_b}^{t_a} \left| \mathbf{v}(\tau)^T \mathbf{w}(\tau) \right| d\tau \leq \phi_1. \tag{6.2.5}$$

The timed attitude constraint could be used to satisfy the requirements of a sensor that needs to avoid being exposed to a bright object for too long.

6.3 Considered Algorithms

The constrained attitude control problem has been investigated through a diverse array of tactics, each with strengths and drawbacks. Obtaining a single al-

gorithm that handles all the spacecraft attitude constraints is difficult. The following methods are some of the most often cited in reference publications.

6.3.1 Augmented Potential Function Method

One of the first methods for solving the constrained attitude control problem comes from McInnes [44]. McInnes approached the problem by manipulating Lyapunov's 2^{nd} method for determining a region of stability of a nonlinear dynamic system. A potential function is first created to drive the current attitude to the desired attitude using Lyapunov's 2^{nd}method. Next, attitude constraints are implemented by creating regions of high potential corresponding to forbidden regions. These high potential regions are superimposed on the original potential function. McInnes superimposes the forbidden zone potentials upon the artificial potential,

$$V = \frac{1}{2}\sum_{i=1}^{3} I_i \omega_i^2 + \frac{1}{2}\sum_{i=1}^{3} \lambda_i \left(\theta_i - \tilde{\theta}_i\right)^2 + \sum \alpha_j \exp\left\{-\beta \sum_{i=1}^{3} \left(\theta_i - \hat{\theta}_i\right)^2\right\}. \qquad (6.3.1)$$

The current attitude is represented by the components Euler angles θ_i , the rotation rate is ω_i, and the principal moments of inertia are I_i for $i = 1,2,3$. The desired attitude is represented by $\tilde{\theta}_i$ and the forbidden attitude is denoted with $\hat{\theta}_i$. From this potential, a control torque can be constructed to ensure that V is negative semi-definite,

$$T_i = -K\omega_i - \sum_{j=1}^{3} G_{ij}^T \lambda_j \left(\theta_j - \tilde{\theta}_j\right) + 2\alpha\beta \sum_{j=1}^{3} G_{ij}^T \left(\theta_j - \tilde{\theta}_j\right) \exp\left\{-\beta \sum_{i=1}^{3} \left(\theta_i - \hat{\theta}_i\right)^2\right\}. \qquad (6.3.2)$$

The shape of the artificial potential is adjusted using the parameters λ. Effectively, this controls the torque amplitude. The shape of the keep out region potential is controlled with α and β. The potential approach is elegant, simple, and easily implemented on an embedded computer for real-time operation.

However, the potential method has a few significant drawbacks. Mainly, the method is not easily expanded to an arbitrary number and type of constraints. In fact, the only constraint that this system can effectively incorporate is the forbidden direction by a single body-fixed boresight vector. As a result, the torque, wheel speed, and body rate constraints are not incorporated within the same framework. They would need to be enforced by a secondary algorithm.

This method was not chosen for the GN&C module due to the limitations in its capabilities. The system would not provide users enough freedom to satisfy common spacecraft attitude requirements without implementing their own additions to the algorithm. A patchwork of solutions could result in one algorithm trying satisfy its constraint and in the process causing another algorithm to create an undesired behavior.

6.3.2 Geometric Algorithms

Moving towards a slightly more complex approach, Hablani develops a geometric approach to avoidance of bright objects.[45] His approach entails calculating ideal tangential paths around sun exclusion cones as seen in Figure 6.3.1. Hablani also extends his algorithm to include constraints that maintain antenna pointing with ground stations. The geometric approach was not pursued because of the increasing complexity in implementation of more than one type and number of constraint.

Simply put, the system does not scale well. Furthermore, there is no innate way to include other constraints, such as body rate constraints, and maximum torque constraints within the same framework. Again, a patchwork of solutions would need to be implemented for each constraint.

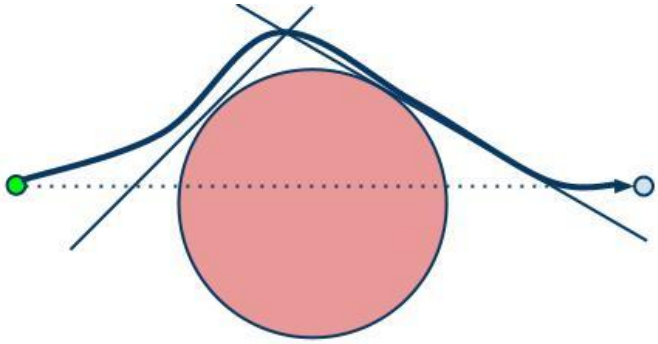


Figure 6.3.1: Geometric approach for constrained attitude control produces a intermediate target through tangent intersections.

6.3.3 Randomized Planning Algorithm

Another method that has been utilized is shown by Frazzoli, and Garcia.[46, 47] In order to include more types of constraints, they use a randomized planning algorithm. A Probabilistic Road Map planner algorithm can provide probabilistic guarantees that convergence will occur to an optimal feasible path within the influence of multiple constraints of different types, given that a feasible path exists. The algorithms operate by constructing graphs of feasible trajectories as seen in Figure 6.3.2. Subsequently, these trajectories are optimized using a smoother algorithm. However, the probabilistic guarantee that a feasible path is found is a function of the number of branches computed.

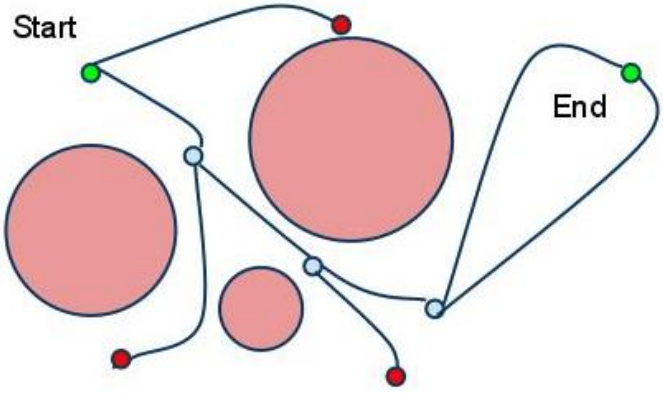


Figure 6.3.2: The randomized planning approach creates a tree of pathways testing for feasibility.

The benefit of the randomized planning approach is that it can satisfy many types of constraints within the same algorithm. These methods can also be applied to groups of spacecraft in 6 degree-of-freedom motions with constraints in relative position and attitude. However, the randomized planner algorithm requires significant computation resources and is complicated to implement. Frazzoli reports that a Pentium II running at 300 MHz was capable of finding a solution trajectory in approximately 1.1 seconds to a simple problem. The control component would need to be conducted by a secondary control algorithm that drives the state to the results produced from the trajectory planner guidance algorithm. The randomized planning method was not pursued due to the complexity of implementation. Instead, another sort of numerical optimization was pursued.

6.4 The Semi-definite Programming Method

A surprisingly succinct and easy to implement alternative to the discussed methods is presented by Kim in a series of papers.[48, 49, 50] Kim utilizes a form of numerical optimization called semi-definite programming (SDP). SDP allows for optimization of a specific subset of convex problems. The equations that describe the constrained attitude control problem in their generic format are non-convex. Non-convex problems have multiple feasible regions and multiple local minima. These characteristics cause numerical optimization routines to experience poor efficiency. Kim develops techniques to convert the non-convex representation into a convex problem. A convex problem has the benefit that there is only one globally optimal solution. Efficient optimizers exist for convex problems.

In SDP, the optimization problem is stated as a set of linear objective functionals and constraints expressed as linear matrix inequalities (LMI). An LMI is an inequality over matrices, interpreted with respect to the positive semi-definite ordering. The challenge is to express the problem in a manner that allows for implementation via SDP, it turns out quaternions representations of attitude can be leveraged to transform the non-convex constrained attitude problem into a convex problem.

The keep out region constraint is often represented with an instrument boresight vector $\mathbf{v}_B(t)$ in the body frame, and an inertial vector $\mathbf{w}$ to the forbidden direction. Note that the vector $\mathbf{w}$ does not vary with time. There is an understood limit that θ, the minimum separation angle between $\mathbf{v}_B$ and $\mathbf{w}$ must be less than or equal to 90°; the result is the common format of the constraint,

$$\mathbf{v}_B(t)^T \mathbf{w} \leq \cos\theta. \tag{6.4.1}$$

A simple equation to relate a body vector to an inertial vector is through a quaternion based rotation,

$$\mathbf{v}_I(t) = \mathbf{v}_B - 2\left(\mathbf{q}^T\mathbf{q}\right)\mathbf{v}_B + 2\left(\mathbf{q}^T\mathbf{v}_B\right)q + 2q\left(\mathbf{v}_B \times \mathbf{q}\right). \tag{6.4.2}$$

Combining (6.4.2) and (6.4.1) leads to the equation

$$\bar{\mathbf{q}}(t)^T \tilde{\mathbf{A}} \bar{\mathbf{q}}(t) \leq 0. \tag{6.4.3}$$

$\tilde{\mathbf{A}}$ is represented as the matrix

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & \mathbf{d} \end{bmatrix}_{4\times4} = \begin{bmatrix} \mathbf{v}_B\mathbf{w}^T + \mathbf{w}\mathbf{v}_B^T - \left(\mathbf{v}_B^T\mathbf{w} + \cos\theta\right)\mathbf{I}_{3\times3} & \mathbf{w}\times\mathbf{v}_B \\ \left[\mathbf{w}\times\mathbf{v}_B\right]^T & \mathbf{v}_B^T\mathbf{w} - \cos\theta \end{bmatrix}. \tag{6.4.4}$$

$\tilde{\mathbf{A}}$ is not positive semi-definite in its current form; therefore, it is not convex with respect to $\bar{\mathbf{q}}$. However, Kim [48, 49, 50] proves that the constrained attitude control problem can be represented in a convex manner using the quaternion's unity norm constraint.

6.4.1 Formulation into a Control Algorithm

The practical portion of the SDP based constrained attitude control algorithm developed by Kim.[48, 49, 50] We minimize a scalar parameter $\alpha(k)$ and

control a set of a set of slack variables $\mathbf{x}(k)$, where k is the time iterate. The $\mathbf{x}(k)$ is a vector that includes the control torques, body rate, and spacecraft quaternion.

$$\min_{\mathbf{x}(k)} \alpha(k). \tag{6.4.5}$$

The linear matrix inequality that constrains the spacecraft from pointing the boresight vector of instrument i into the corresponding exclusion region, is given by

$$\mathbf{x}(k)^T \left\{\mathbf{H}^T \tilde{\mathbf{A}}_i \mathbf{H}\right\} \mathbf{x}(k) \leq 0 \tag{6.4.6}$$

which yields [50]

$$\boxed{\begin{bmatrix} \mu_i & (\mathbf{H}\mathbf{x}(k))^T \\ \mathbf{H}\mathbf{x}(k) & \left(\mu_i \mathbf{I}_4 + \tilde{\mathbf{A}}_i\right)^{-1} \end{bmatrix} \geq 0, \ i = 1,2,...,m}. \tag{6.4.7}$$

Where, $\tilde{\mathbf{A}}_i$ is defined in (6.4.4), and μ_i is chosen to be strictly greater than the largest eigenvalue of $-\tilde{\mathbf{A}}_i$. Additionally,

$$\mathbf{x} = \begin{bmatrix} \mathbf{u}(k) \\ \boldsymbol{\omega}(k+1) \\ \bar{\mathbf{q}}(k+2) \end{bmatrix}_{10\times 1}. \tag{6.4.8}$$

And the $\mathbf{H}$ matrix only serves to isolate the quaternion from the slack variables,

$$\mathbf{H} = \begin{bmatrix} \mathbf{0}_{4\times 6} & \mathbf{I}_{4\times 4} \end{bmatrix}. \tag{6.4.9}$$

Another linear matrix inequality is formed to drive the current quaternion to the desired quaternion:[50]

$$\boxed{\begin{bmatrix} \alpha & \left(\mathbf{E}(k)^{1/2}\begin{bmatrix}\mathbf{x}(k)\\1\end{bmatrix}\right)^T \\ \mathbf{E}(k)^{1/2}\begin{bmatrix}\mathbf{x}(k)\\1\end{bmatrix} & \mathbf{I}_{11}\end{bmatrix} \geq 0} \tag{6.4.10}$$

Where the parameter α that is minimized, drives the quaternion error towards zero. Defining,

$$\mathbf{E}(k) = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times4} & \boldsymbol{\omega} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \mathbf{0}_{4\times4} & \mathbf{0}_{4\times1} \\ \mathbf{0}_{1\times3} & \boldsymbol{\omega}^T & \mathbf{0}_{1\times4} & \boldsymbol{\omega}^T\boldsymbol{\omega} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{6\times6} & \mathbf{0}_{6\times4} & \mathbf{0}_{6\times1} \\ \mathbf{0}_{4\times6} & \mathbf{Q}^T\mathbf{Q} & \mathbf{0}_{4\times1} \\ \mathbf{0}_{1\times6} & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times1} \end{bmatrix}. \tag{6.4.11}$$

Here the matrix $\mathbf{Q}$ is composed of the elements of the desired quaternion,

$$\mathbf{Q} = \begin{bmatrix} q_{4d} & q_{3d} & -q_{2d} & -q_{1d} \\ -q_{3d} & q_{4d} & q_{1d} & -q_{2d} \\ q_{2d} & -q_{1d} & q_{4d} & -q_{3d} \\ q_{1d} & q_{2d} & q_{3d} & q_{4d} \end{bmatrix}. \tag{6.4.12}$$

Kinematic constraints are satisfied with the equality constraint, thus forcing the $\mathbf{x}$ vector to be a kinematically feasible motion for each iteration [50]

$$\boxed{\mathbf{F}(k)\mathbf{x}(k) = \mathbf{y}(k)}. \tag{6.4.13}$$

Where,

$$\mathbf{y}(k) = \begin{bmatrix} J_1\omega_1(k) + \Delta t\,(J_2 - J_3)\,\omega_2(k)\,\omega_3(k) \\ J_2\omega_2(k) + \Delta t\,(J_3 - J_1)\,\omega_3(k)\,\omega_1(k) \\ J_3\omega_3(k) + \Delta t\,(J_1 - J_2)\,\omega_1(k)\,\omega_2(k) \\ \bar{\mathbf{q}}(k+1) \end{bmatrix} \tag{6.4.14}$$

$$\mathbf{F}(k) = \begin{bmatrix} -\Delta t \mathbf{I}_3 & \mathbf{J} & \mathbf{0}_{3\times 4} \\ \mathbf{0}_{4\times 3} & \mathbf{R}(k+1) & \mathbf{I}_4 \end{bmatrix} \tag{6.4.15}$$

$$R(k) = \frac{\Delta t}{2} \begin{bmatrix} -q_4(k) & q_3(k) & -q_2(k) \\ -q_3(k) & -q_4(k) & q_1(k) \\ q_2(k) & -q_1(k) & -q_4(k) \\ q_1(k) & q_2(k) & q_3(k) \end{bmatrix} \tag{6.4.16}$$

The spacecraft angular velocity constraint is given by [50]

$$\boxed{|\boldsymbol{\omega}(k)| \leq \boldsymbol{\omega}_{Max}}. \tag{6.4.17}$$

Finally, the torque constraint is given by [50]

$$\boxed{|\mathbf{u}(k)| \leq \mathbf{u}_{Max}}. \tag{6.4.18}$$

The boxed constraints are are given to the SDP algorithm. For each iteration, the SDP algorithm finds the values of $\mathbf{x}$ that satisfy the inequality and equality constraints to within some numerical tolerance which is adjusted by the user.

6.4.2 SDP Controller Implementation

In order to utilize the semi-definite programming based constrained attitude control algorithm, a semi-definite programming solver is required. Multiple open source clients for semi-definite programming exist. For example, SeDuMi, SDPA, and CSDP. Kim implements the algorithm using SeDuMi in MATLAB, but this is not feasible for an embedded computer in a CubeSat.[50] The challenge then is to translate the mathematical notation to the particular input format of an acceptable

solver. The feature set and input format of each solver is not consistent. Some solvers are better for very specific subsets of the SDP problem. For this project, the CSDP solver, written by Borchers, was chosen because it was written in C and designed for performance.[51]

CSDP considers the semi-definite programming problem in a form that is not directly compatible with the representation formed by Kim. Therefore, a translation needs to be completed. Fortunately, a user-friendly tool for translating the mathematical notation into the a representation that CSDP accepts exists. A software package called "Yet Another Linear Matrix Inequality Parser" (YALMIP) is capable of converting a problem written in mathematical notation into the input format of the CSDP algorithm.[52]

The driving motivation behind investigating the SDP constrained attitude control approach was to test the feasibility of embedded utilization on a CubeSat. To that end, the code was implemented onto a MPC5200 running a generic Linux kernel operating at 400 MHz with 32 MB RAM shown in Figure 6.4.1. While the CSDP solver is portable, it was not designed with embedded systems in mind. Nevertheless, the modifications required to the source code were not insurmountable and generally focused on the Linear Algebra Pack (LAPACK) and Basic Linear Algebra Subroutines (BLAS) interfaces.

Figure 6.4.1: MPX5200 single board computer from Microsys. [Photo Credit: Henri Kjellberg]

CSDP relies heavily on linear algebra operations provided by BLAS and LAPACK. In fact, most of the computation time is spent in the linear algebra subroutines. BLAS and LAPACK were originally implemented in Fortran and require a Fortran compiler and run-time library to operate. The MPC5200 which has a Power PC 603 architecture does not come with a pre-built Fortran compiler and run-time library. Furthermore, none are available in the open source community. A fruitless attempt was made to compile a Fortran compiler for the 603 architecture. Instead, BLAS and LAPACK calls in the CSDP source code were modified to utilize the C version of BLAS provided in the GNU Scientific Library and CLAPACK. CLAPACK is a computer generated Fortran to C translation of the unoptimized implementation of LAPACK.[53]

The modified software was tested with the standard CSDP test libraries and found to operate as expected. However, not being able to use the highly optimized

Fortran subroutines caused some problems. The subroutines implemented onto the MPC5200 are far from optimal. Naive implementations of BLAS and LAPACK have the potential of being an order of magnitude slower than optimized versions. To that end, future work should focus on implementing an optimized BLAS routine such as the Automatically Tuned Linear Algebra Software (ATLAS) onto the Power PC 603 architecture. As the number of constraints increases, the computation time increases. Thus, any improvements in the run-time efficiency will allow for more constraints to be implemented.[54]

6.4.3 SDP Controller Performance

For a simulation with one hard pointing constraint, Figure 6.4.2 shows the output of each iteration of the optimization routine as it ran on the MPC5200. Typically, the constrained attitude control problem is solved in less than 15 iterations. For Bevo-2, the controller is required to compute a torque command at a rate faster than 5 Hz. Approximately 0.1 seconds are allowed between the retrieval of the best estimate navigation solution to transmission of control commands to the actuators. Even with the unoptimized implementations linear algebra subroutines, the code runs fast enough to calculate a solution in real-time. It takes 0.0875 seconds on average to calculate a full solution. The computation time is a conservative estimate because the input to the CSDP algorithm is read as a file via Ethernet from the host PC hard drive. On a flight system, the input data would be stored in RAM.

In addition to the control algorithm, the flight computer must also communicate with all the sensors and actuators, communicate with the host spacecraft command and data handling computer, and calculate a navigation solution. A controller

```
Iter:  0 Ap:  0.00e+00 Pobj:  -4.1148768e+03 Ad:  0.00e+00 Dobj:  0.0000000e+00
Iter:  1 Ap:  9.60e-01 Pobj:  -3.7397853e+03 Ad:  8.73e-01 Dobj:  1.8089533e+01
Iter:  2 Ap:  7.84e-01 Pobj:  -1.8314124e+03 Ad:  9.07e-01 Dobj:  3.5513715e+01
Iter:  3 Ap:  9.57e-01 Pobj:  -3.1038923e+02 Ad:  8.67e-01 Dobj:  3.7326575e+01
Iter:  4 Ap:  1.00e+00 Pobj:  -2.6216505e+01 Ad:  9.26e-01 Dobj:  2.2243699e+01
Iter:  5 Ap:  1.00e+00 Pobj:  -3.0037770e+00 Ad:  9.19e-01 Dobj:  4.7302105e+00
Iter:  6 Ap:  1.00e+00 Pobj:  6.7607855e-01 Ad:  9.19e-01 Dobj:  1.5006705e+00
Iter:  7 Ap:  1.00e+00 Pobj:  9.5899519e-01 Ad:  9.37e-01 Dobj:  1.0585133e+00
Iter:  8 Ap:  1.00e+00 Pobj:  9.9412777e-01 Ad:  9.24e-01 Dobj:  1.0266784e+00
Iter:  9 Ap:  1.00e+00 Pobj:  1.0152339e+00 Ad:  1.00e+00 Dobj:  1.0190345e+00
Iter:  10 Ap:  1.00e+00 Pobj:  1.0178501e+00 Ad:  9.81e-01 Dobj:  1.0180484e+00
Iter:  11 Ap:  9.99e-01 Pobj:  1.0179713e+00 Ad:  1.00e+00 Dobj:  1.0179902e+00
Iter:  12 Ap:  1.00e+00 Pobj:  1.0179835e+00 Ad:  1.00e+00 Dobj:  1.0179858e+00
Iter:  13 Ap:  1.00e+00 Pobj:  1.0179870e+00 Ad:  1.00e+00 Dobj:  1.0179870e+00
Iter:  14 Ap:  9.60e-01 Pobj:  1.0179872e+00 Ad:  9.59e-01 Dobj:  1.0179872e+00
Success:  SDP solved
Elements time:  0.002579
Factor time:  0.001030
Other time:  0.083948
Total time:  0.087557
```

Figure 6.4.2: Output from the CSDP implementation solving an iteration of the constrained attitude control problem.

that takes 0.0875 seconds to execute should leave enough computation margin for these additional tasks. Furthermore, there is room for significant performance improvements in the linear algebra subroutines.

Simulations were run using the StarBox software environment connected to the embedded system via an RS-232 cable. The spacecraft was modeled after Bevo-2, with a single sensitive sensor aligned with the x-axis in the body reference frame. First, an unconstrained rotation is shown in Figure 6.4.3. Here, the boresight vector of the sensitive sensor is shown as vectors and as a line projected onto an invisible unit sphere. The motion is what is expected, an eigenaxis rotation.

Next, in Figure 6.4.4, the same rotation is shown, except now with a 45 degree exclusion zone around the sun vector $w = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, shown in magenta.

Figure 6.4.5 shows a plot of the attitude constraint as a function of time. The 60 second slew spends approximately 15 seconds on the edge of the exclusion zone.

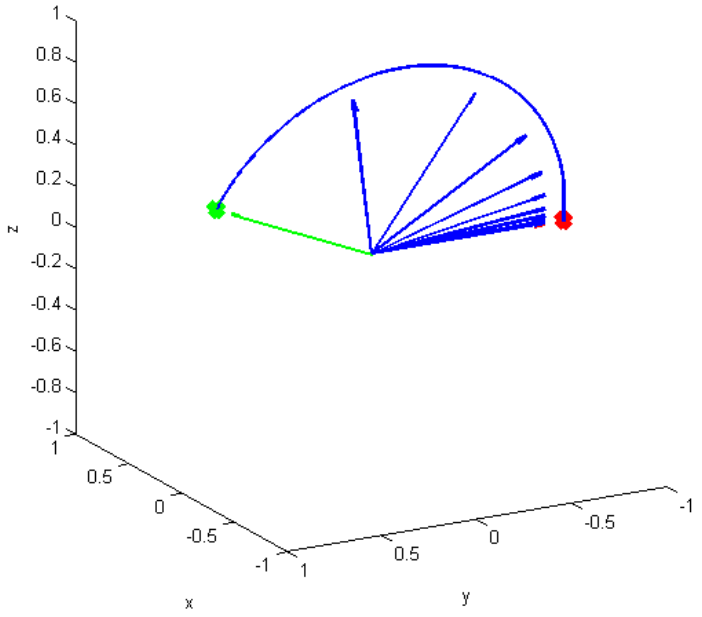


Figure 6.4.3: Sensor boresight vector projected onto the units sphere in the unconstrained eigenaxis attitude slew. The motion begins at the green point and ends at the red point.

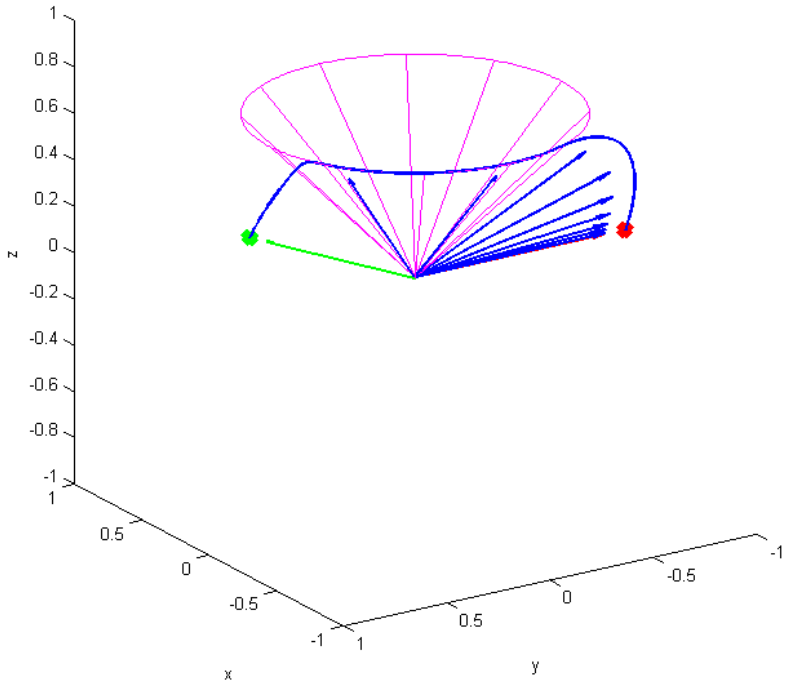


Figure 6.4.4: Constrained attitude slew with the constraint cone in magenta. The sensitive sensor boresight vector evolves on the sphere.

The attitude constraint is satisfied as long $\bar{\mathbf{q}}(t)^T \tilde{\mathbf{A}} \bar{\mathbf{q}}(t) \leq 0$, where $\tilde{\mathbf{A}}$ is a function of the sensor boresight vector in the body frame, the bright object in the inertial frame, and the angular size of the exclusion zone described in (6.4.4).

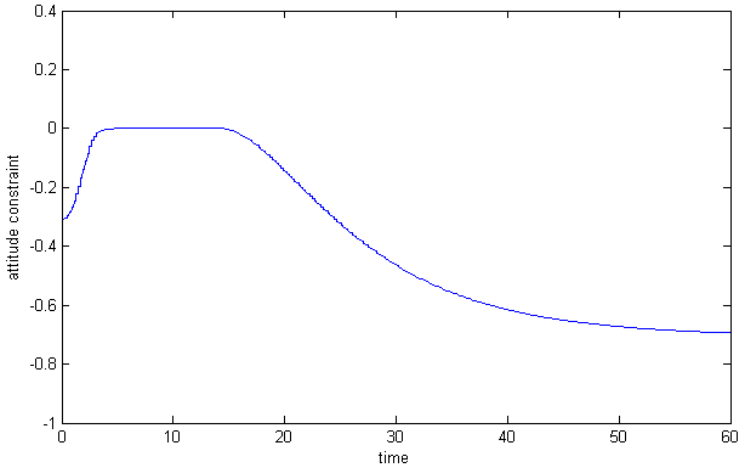


Figure 6.4.5: The attitude constraint is satisfied as long as the value is negative.

Figure 6.4.6 shows the resultant reaction wheel attitude control torques and the resultant angular momentum as a function of time. The simulation includes sensor and actuator noise, therefore they are not perfectly smooth. Despite navigation and control noise, the controller produces an efficient slew. The reaction wheel maximum torque capabilities are satisfied inherently through the SDP algorithm, thus no saturation command is required. For this simulation, the reaction wheel torques were limited to 1 mNm-s, corresponding to the Sinclair Interplanetary 10 mNm-s reaction wheels. Future investigations should pursue implementing integral constraints to limit the reaction wheel angular momentum (corresponding to wheel speeds).

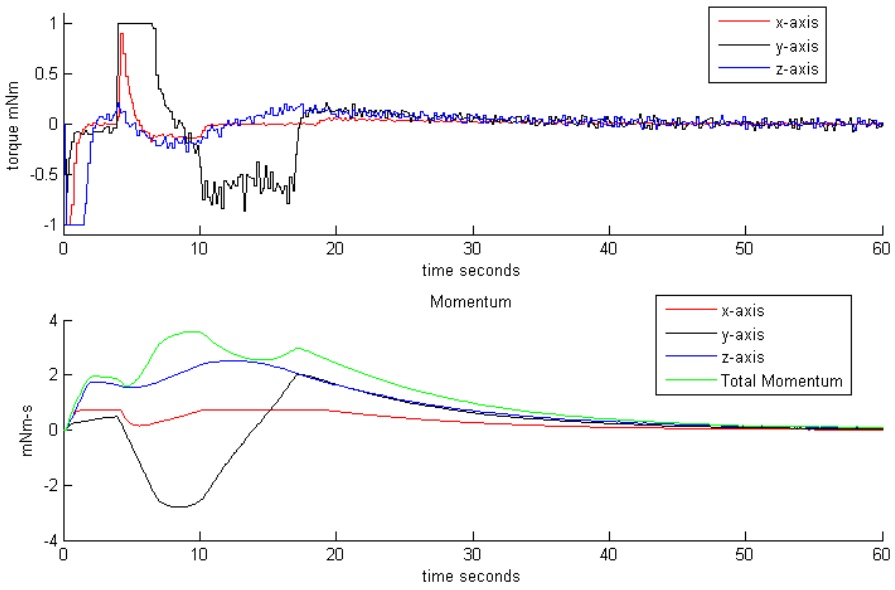


Figure 6.4.6: Reaction wheel control torques and resultant stored angular momentum.

Figure 6.4.7 shows the spacecraft body rates in degrees per second. Body rate constraints are often imposed for spacecraft in order to maintain observability. Angular rate sensors, as well as sun sensors can obtain incorrect measurements if the spacecraft platform is spinning too fast. An angular rate maximum of 5 degrees/second was imposed for this simulation. The pitch rate reaches this constraint twice during the reorientation maneuver. Note that these constraints are satisfied per axis. In comparison with the basic constrained body rate quaternion controller presented in Chapter 6, the SDP controller allows for significantly more aggressive maneuvers because the body rate constraints are not satisfied as an aggregate, but individually. Therefore, should it be found optimal by the controller, each axis of the body could be rotating at the maximum (5 degrees/second in this case).

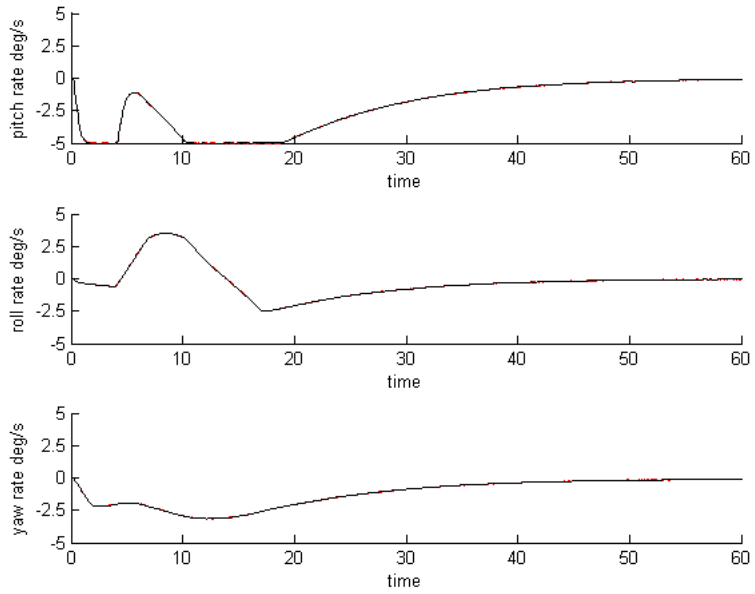


Figure 6.4.7: Constrained pitch, roll, and yaw rotation rates.

6.4.4 Advantages of the SDP Controller

The SDP constrained attitude controller has been shown to be feasible in an embedded system operating on a computationally constrained CubeSat. The algorithm can efficiently constrain the boresight vector of a sensitive sensor to avoid rotating into an exclusion zone. Furthermore, the algorithm allows for torque and body rate constraints to be satisfied within the same algorithm. It is straightforward to implement the algorithm, much of the optimization software provided is stable and professional releases are available through the open source community.

Utilizing a constrained attitude controller can allow for design decisions to decrease the cost of a spacecraft mission. For example, fewer sun sensors need to

be purchased and incorporated onto a spacecraft, but the motion can be constrained to always maintain line of sight to the sun for the available sensors, thus money, volume, and power can be saved. The SDP based constrained attitude controller can be expanded to an arbitrary number of time invariant constraints, however with each additional constraint the computation time increases. Through further optimization, the controller presented here could be implemented onto the Bevo-2 spacecraft in order to enable greater autonomy in the spacecraft's reorientation maneuvers.

Chapter 7

Implementation in Hardware

The transition from paper design and analysis is important, but filled with risk. Complex spacecraft components such as sun sensors and reaction wheels generally have costs on the order of thousands of dollars. Therefore, it is difficult to decide when one is completed with the design phase and can move on to the prototyping phase. Nevertheless, after the design has reached a certain point progress can only be made when components are purchased and prototyping begins. This chapter shows the iterative transition from design analysis to implementation in hardware.

7.1 Hardware-in-the-Loop Simulation

The transition from simulation to implementation does not necessarily need to occur in one step. Instead of implementing a hardware equivalent of what has been built in simulation, individual components can be attached to StarBox to create a hardware-in-the-loop test environment. In particular, this is useful for the development and testing of the flight software on the embedded flight computer. Figure 7.1.1 shows a block diagram of the hardware-in-the-loop configuration of the StarBox for testing the flight software. Here, the embedded flight computer performs the state estimation from simulated measurements and produces a control command to simulated actuators. The data is transferred via RS-232 in order to provide for the

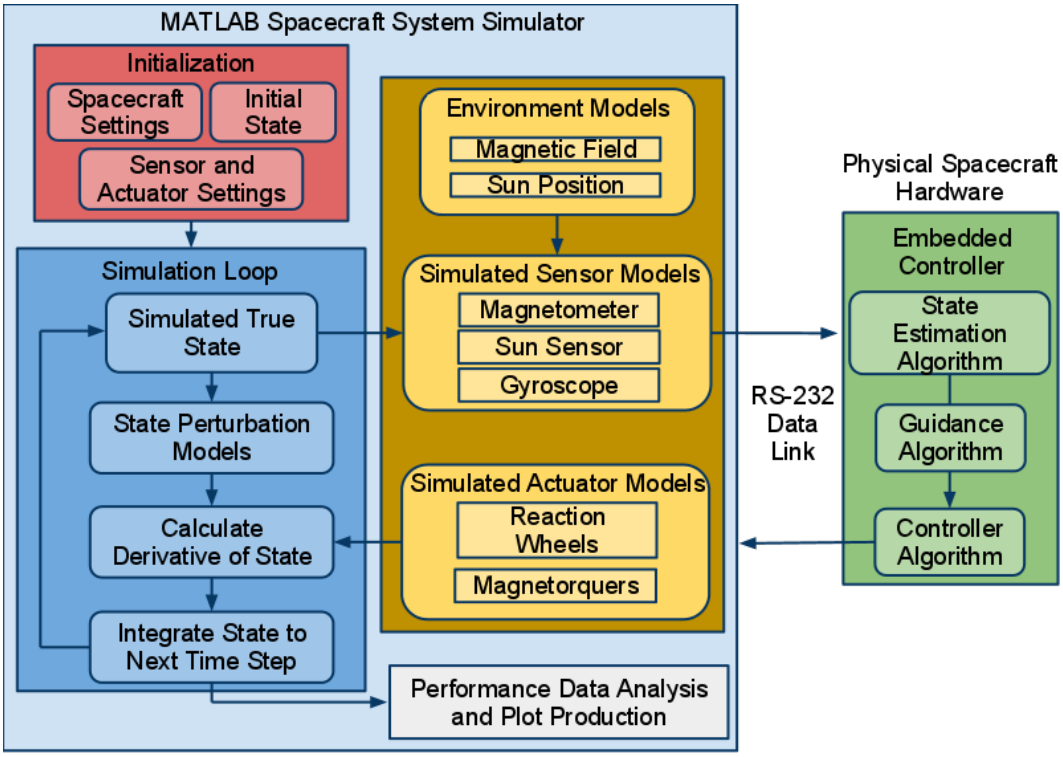
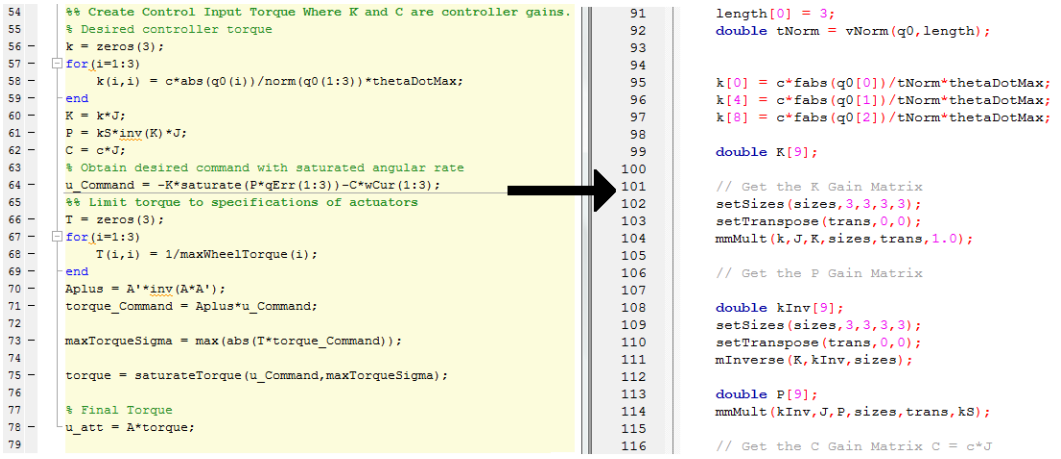


Figure 7.1.1: Block diagram of the hardware-in-the-loop configuration of StarBox. Instead of performing the GN&C computation on the computer running StarBox as in Figure 3.0.1, algorithms are executed on the spacecraft embedded computer.

opportunity to include communication delays in the overall computation budget.

The flight software is designed and developed in MATLAB. Utilizing MATLAB for the design allows for quick prototyping because the syntax is very close to the mathematical notation of the navigation and control algorithms. The hardware-in-the-loop configuration is excellent for verifying the integrity of the flight software because the output of the MATLAB and C implementations can be compared with each other directly with the same input in real-time. Using this comparative process, the navigation and control implementations were translated from MAT-



```
%% Create Control Input Torque Where K and C are controller gains.
% Desired controller torque
k = zeros(3);
for(i=1:3)
    k(i,i) = c*abs(q0(i))/norm(q0(1:3))*thetaDotMax;
end
K = k*J;
P = kS*inv(K)*J;
C = c*J;
% Obtain desired command with saturated angular rate
u_Command = -K*saturate(P*qErr(1:3))-C*wCur(1:3);
%% Limit torque to specifications of actuators
T = zeros(3);
for(i=1:3)
    T(i,i) = 1/maxWheelTorque(i);
end
Aplus = A'*inv(A*A');
torque_Command = Aplus*u_Command;

maxTorqueSigma = max(abs(T*torque_Command));

torque = saturateTorque(u_Command,maxTorqueSigma);

% Final Torque
u_att = A*torque;
```

```
length[0] = 3;
double tNorm = vNorm(q0,length);


k[0] = c*fabs(q0[0])/tNorm*thetaDotMax;
k[4] = c*fabs(q0[1])/tNorm*thetaDotMax;
k[8] = c*fabs(q0[2])/tNorm*thetaDotMax;

double K[9];

// Get the K Gain Matrix
setSizes(sizes,3,3,3,3);
setTranspose(trans,0,0);
mmMult(k,J,K,sizes,trans,1.0);

// Get the P Gain Matrix

double kInv[9];
setSizes(sizes,3,3,3,3);
setTranspose(trans,0,0);
mInverse(K,kInv,sizes);

double P[9];
mmMult(kInv,J,P,sizes,trans,kS);

// Get the C Gain Matrix C = c*J
```

Figure 7.1.2: Transition from flight software design implementation in MATLAB to C.

LAB to C as seen in Figure 7.1.2.

7.2 GN&C Module Hardware Design

Using SolidWorks, a computer aided design (CAD) of the GN&C module was updated with all the possible component choices. The way that the components fit is important in reducing the total volume consumption of the GN&C module. The volume allocated for the GN&C module was that of a 1-U CubeSat leaving the remaining 2-U for the rest of the spacecraft. No external sensors or deployables were allowed; all the necessary hardware needed to fit within the confines of the GN&C module envelope. All of the computation required for attitude determination and control are performed on the embedded computer that is integrated inside the module. Figure 7.2.1 shows a block diagram of the hardware with electrical connections to the sensors and actuators. Figure 7.2.2 shows a CAD of the GN&C module with an optional add-on cold gas thruster. The entire module with thruster

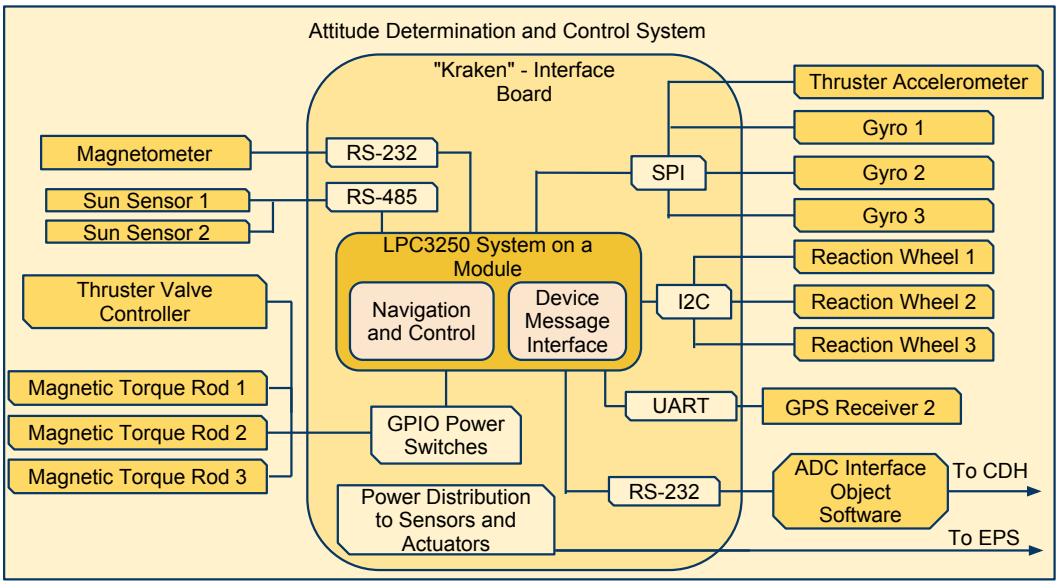


Figure 7.2.1: GN&C module hardware block diagram.

provides 6 degree-of-freedom control to the host spacecraft. The design takes into account the placement of fasteners and the routing of wires to the connectors of all the sensors and actuators. Figure 7.2.3 shows the module attached to the 3-U CubeSat ARMADILLO.

7.3 GN&C Electrical Interface Design

The LPC3250 system on a module requires a host circuit board to expose all the electrical communication interfaces to the sensors and actuators. Figure 7.3.1 shows the block diagram of the interface board. This board must adjust the electrical signals to the correct standards for each device. For example, the UART 2 must be driven at RS-232 for the magnetometer and the UART 3 must be level converted for connecting to a GPS receiver. Using the specifications for the GN&C module

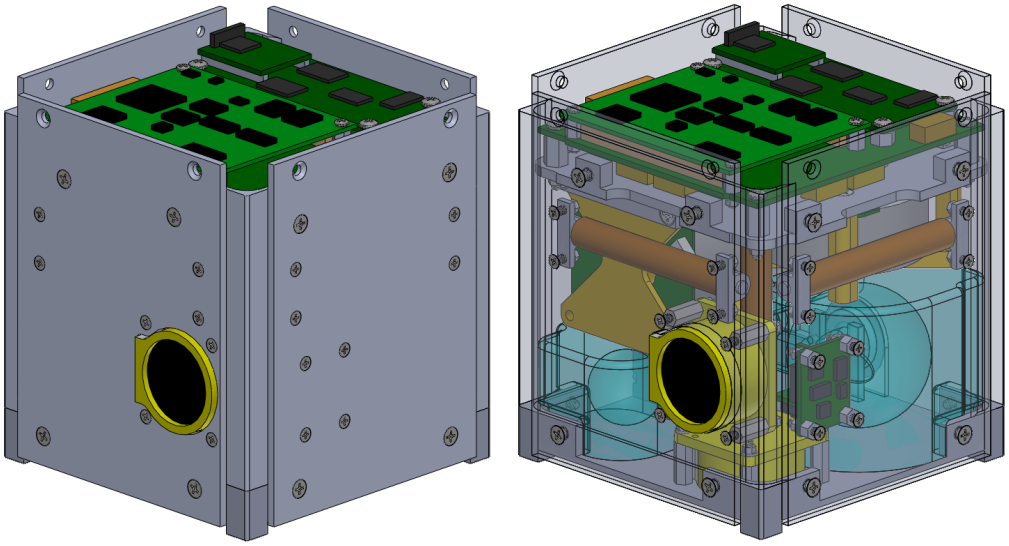


Figure 7.2.2: Computer model of the bolt-on GN&C module and its components.

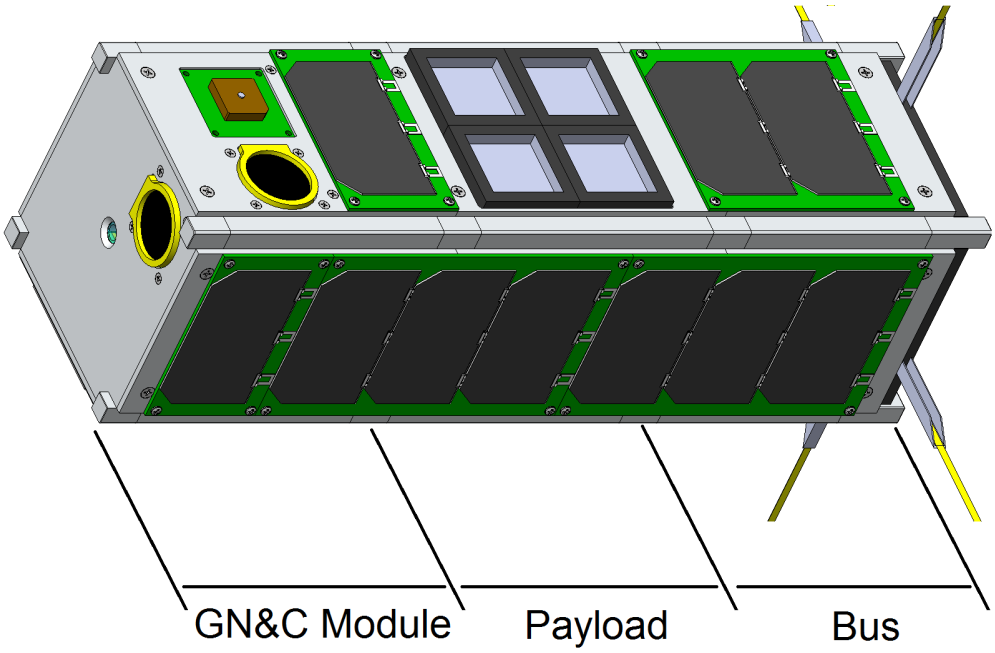


Figure 7.2.3: The GN&C module can be bolted on to the end of a typical 3-U CubeSat.

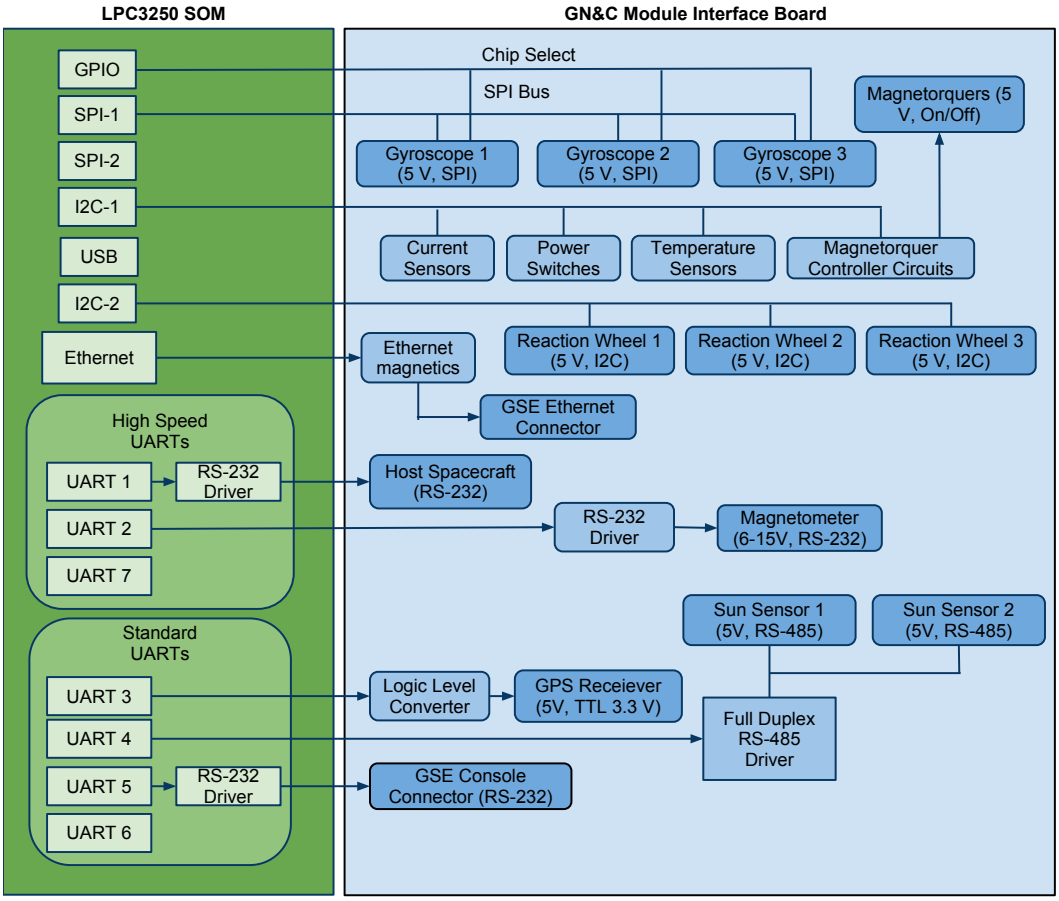


Figure 7.3.1: Block diagram of the electrical interface between the flight computer and the peripherals.

sensors and actuators, a circuit board was professionally designed, fabricated, and assembled. The result is shown in Figure 7.3.2.

7.4 Work Required to Prepare for Flight

At the conclusion of this thesis, the GN&C module has completed the prototype phase. Utilizing the prototype test bench the flight software can be iterated

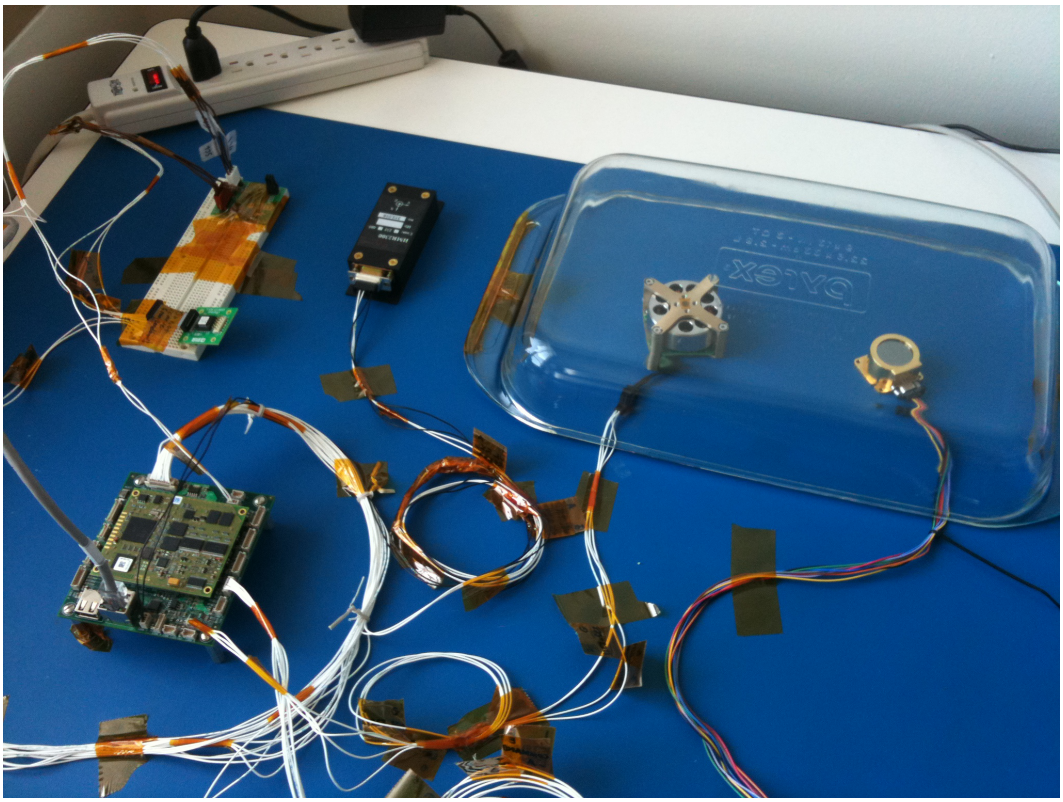


Figure 7.3.2: Prototype GN&C test bench with with LPC3250 connected to sun sensor, reaction wheel, magnetometer, and gyroscope via custom interface board. [Photo Credit: Henri Kjellberg]

until the code is stable. Particular focus will be placed on the subtle details of the GN&C software architecture. These details include internal error handling, data management, and host spacecraft communications. Using the prototype test bench, day-in-the-life tests can be performed using the hardware-in-the-loop simulation to establish the stability of the system over long periods of time.

Next, a fully integrated GN&C module engineering design unit (EDU) in the flight form factor will be fabricated. The transition from bench top prototype to EDU will reveal new issues in the software and hardware interfaces. In particular, the tight integration of electronics will introduce significant electromagnetic interference. One by one the issues that arise will be solved in an iterative process. Simultaneously, the EDU will undergo environmental testing in thermal vacuum chambers and vibration tables in order to prove the mechanical design.

Finally, with the lessons learned from the EDU, two flight versions of the GN&C module will be fabricated. One will be delivered to the ARMADILLO CubeSat for the UNP-7 flight competition review (FCR), the other will be integrated into the Bevo-2 spacecraft for delivery to NASA for the LONESTAR-2 mission. Both deliveries occur in August 2012.

Chapter 8

Conclusion

CubeSats offer an opportunity to provide a new way to conduct science in space. However, CubeSats are highly constrained in their performance due to a combination of requirements for low power, low actuation, low volume, and low mass. Through innovative solutions such as the design of a versatile GN&C module with advanced constrained control methods as shown in this thesis, new categories of missions can begin to be realized. In the future, large constellations of cooperative picosatellites flying in formation with complex pointing and formation requirements will perform fascinating science, commercial, and defense missions. As the number of spacecraft increases, the ability to directly control individual spacecraft in the constellation decreases. Therefore, these spacecraft must become capable of satisfying their own state constraints with greater autonomy. This thesis is a starting point for the work ahead. The analysis provided here indicates that a combination of commercial-off-the-shelf hardware, and standard navigation, and control algorithms can be combined together to create a GN&C module that can meet the requirements of various scientific CubeSat missions. The GN&C module is currently being fabricated and is intended to fly on the UT-Austin designed Bevo-2 and ARMADILLO CubeSats in the years following the publication of this thesis.

Bibliography

[1] Weselby, C., "PharmaSat," Internet, July 2009, NASA Ames Research Center: http://www.nasa.gov/mission_pages/smallsats/pharmasat/main/index.html

[2] Jorgensen, T., "CubeSat-based Science Missions for Space Weather and Atmospheric Research," Internet, July 2006, National Science Foundation: http://www.nsf.gov/pubs/2010/nsf10537/nsf10537.htm.

[3] Munakata, R., *CubeSat Design Specification*, Revision 12, The CubeSat Program, Cal Poly SLO, August 2009.

[4] Beasley, C., "PharmaSat Assembly," Internet. May 2009, NASA Ames Research Center, http://www.nasa.gov/mission_pages/smallsats/pharmasat_assembly.html.

[5] Lightsey, G., "Guidance, Navigation, and Control System for Maneuverable Pico-Satellites," Internet, May 2010, NASA STTR 2008 Proposal Summary: http://sbir.gsfc.nasa.gov/SBIR/abstracts/08/sttr/phase2/STTR-08-2-T6.01-9905.html

[6] Muñoz, S., Greenbaum, J., Lightsey, E., Campbell, T., Stewart, S., and Holt, G., *The FASTRAC Mission: Operations Summary and Preliminary Experiment Results*, 25th AIAA/USU Coneference on Small Satellites, SSC11-II-4, 2011.

[7] Hernandez, L., "Four Satellites Sit Atop a Minotaur IV Launch Vehicle at Kodiak Launch Complex, Alaska, Oct. 18, 2010", U.S. Air Force,

http://defenseimagery.mil/imagery.html.

[8] Bhatti, J., "BEVO1 Operations," Internet, Sept. 2009, The University of Texas at Austin Satellite Design Laboratory:

http://paradigm.ae.utexas.edu/ops.

[9] Kjellberg, H., Brumbaugh, K., Lightsey, G., "Bevo-2 Systems Concept Review," Presentation to the LONESTAR-2 Program, NASA Johnson Space Center, Sept. 10, 2010,

[10] McBryde, C., "ARMADILLO," Internet, July 2011, The University of Texas at Austin Satellite Design Laboratory:

http://armadillo.ae.utexas.edu/.

[11] Tsiotras, P., Shen, H., *Satellite Attitude Control and Power Tracking with Energy/Momentum Wheels*, Journal of Guidance, Control, and Dynamics Vol. 24, No. 1, January–February 2001.

[12] Muñoz, S., Christian, J., Lightsey, E., *Development of an End to End Simulation Tool for Autonomous Cislunar Navigation*, AIAA-2009-5995 AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois, Aug. 10-13, 2009.

[13] Sinclair, D., "End Item Data Package Digital Sun Sensors," Hardware Manual, Sinclair Interplanetary, March 2011.

[14] Enright, J., Sinclair, D., Li, C., *Embedded algorithms for the SS-411 digital sun sensor*, Acta Astronautica, 64 (2009) 906-924.

[15] Maus, S., McLean, S., et. al, *The US/UK World Magnetic Model for 2010-2015*, NOAA Technical Report NESDIS/NGDC, 2010.

[16] Christian, J., Lightsey, E., *Sequential Optimal Attitude Recursion Filter*, Journal of Guidance, Control, and Dynamics 0731-5090 vol.33 no.6 (1787-1800), 2010.

[17] Schalkowsky, S., Harris, M., *Spacecraft Magnetic Torques*, NASA Space Vehicle Design Criteria (Guidance and Control), NASA SP-8018, March 1969.

[18] Garner, Michael D., *Systems Engineering Processes for a Student-based Design Laboratory*, Masters Thesis, The University of Texas at Austin, December 2009.

[19] Larson, Wiley J. and Wertz, James R., *Space Mission Analysis and Design*, Third Edition, Space Technology Series, 1999.

[20] "CubeSat Compact Three-Axis Attitude Actuator and Sensor Pack with Sinclair Interplanetary," Data Sheet, University of Toronto Institute of Aerospace Studies Spaceflight Laboratory, 2009.

[21] "MAI-100 Miniature 3-Axis ADCS Product Specification," Maryland Aerospace, Inc., 2010.

[22] "Smart Digital Magnetometer HMR2300," Data Sheet, Honeywell, 2006.

[23] "Tri-axial Magnetometer," Data Sheet, Satellite Services LTD, 2010.

[24] "Cute-1.7 + APD II Project," Internet, Oct. 2009, Tokyo Institute of Technology Lab for Space Systems:

http://lss.mes.titech.ac.jp/ssp/cute1.7/index_e.html.

[25] "SS-411 Digital Sun Sensor Interface Control Document," Rev. 2.2, Sinclair Interplanetary, 2008.

[26] "Miniaturised Analog Fine Sun Sensor," Data Sheet, Innovative Solutions in Space, 2010.

[27] "Medium Sun Sensor," Data Sheet, Comtech AeroAstro, Inc., 2011.

[28] "Digital Output, High Precision Angular Rate Sensor ADIS16130," Rev. B, Analog Devices, 2010.

[29] "Programmable Digital Gyroscope Sensor ADIS16260/ADIS16265," Rev. C, Analog Devices, 2011.

[30] "Reaction Wheel RW 1 for Pico and Nano Satellites," Datasheet, Astro-und Feinwerktechnik Adlershof GmbH, 2011.

[31] "Interface Control Document - Electrical Reaction Wheel with 4 V Power," Rev. 1.1, Sinclair Interplanetary, 2010.

[32] Sinclair, D., C. C. Grant, R. E. Zee, *Developing, Flying and Evolving a Canadian Microsatellite Reaction Wheel – Lessons Learned*, 15th CASI Astronautics Conference, Toronto, Canada, 2010.

[33] "phyCORE-i.MX31 Hardware Manual," PHYTEC America LLC, 2009.

[34] "phyCORE-LPC3250 System on Module and Carrier Board Hardware Manual," PHYTEC America LLC, 2009.

[35] "User's Manual MPX5200G Rev. 2," MicroSys GmbH, 2007.

[36] Wahba, G., *A Least Square Estimate of Satellite Attitude*, SIAM Review, Vol. 7, No. 3, p. 409, July 1965.

[37] Keat, J., *Analysis of Least-Squares Attitude Determination Routine DOAOP*, Computer Sciences Corp., Rept. CSC/TM-77/6034, Feb. 1977.

[38] Markley, F., and Mortari, D., *Quaternion Attitude Estimation Using Vector Observations*, Journal of hte Astronautical Sciences, Vol. 48, Nos. 2-3, pp. 359-380, April-Sept. 2000.

[39] Shuster, M., and Oh, S., *Three-Axis Attitude Determination from Vector Observations*, Journal of Guidance and Control, Vol, 4, No. 1, pp. 70-77, Jan.-Feb. 1981.

[40] Mortari, D., *ESOQ: A Closed-Form Solution to the Wahba Problem*, Journal of the Astronautical Sciences, Vol. 45, No. 2, pp. 195-204, April-June 1997.

[41] Christian, J., *Autonomous Spacecraft Attittude Estimation with an Optical Sensor*, ASE 381P.8 Stochastic Estimation and Control - Final Project, University of Texas at Austin, 2009.

[42] Wie, B., *Space Vehicle Dynamics and Control*, Second Edition, AIAA Education Series, AIAA, 2008.

[43] Koenig, J.D., *A Novel Attitude Guidance Algorithm for Exclusion Zone Avoidance*, Aerospace conference, 2009 IEEE, pp. 1-10, 7-14 March 2009.

[44] McInnes, C. R., *Large Angle Slew Maneuvers with Autonomous Sun Vector Avoidance*, AIAA Journal of Guidance, Control, and Dynamics, 17(4):875-877, 1994.

[45] Hablani, H., B., *Attitude Commands Avoiding Bright Objects and Maintaining Communications with Ground Station*, AIAA Journal of Guidance, Control, and Dynamcis, 22(6), December 1999.

[46] Frazzoli, E., Dahleh, M. A., Feron, E., and Kornfeld, R., *A Randomized Attitude Slew Planning Algorithm for Autonomous Spacecraft*, Proceedings of the AIAA Guidance Navigation, and Control Conference, AIAA, Montreal, Canada 2001.

[47] Garcia, I., and How, J., *Trajectory Optimization for Satellite Reconfiguration Maneuvers with Position and Attitude Constraints*, Proceedings of the American Control Conference, Portland, Oregon 2005.

[48] Kim, Y., Mesbahi, M.; , *Quadratically Constrained Attitude Control via Semidefinite Programming*, Proceedings 42nd IEEE Conference on Decision and Control, Vol. 4, pp. 3408- 3413, 9-12 Dec. 2003

[49] Kim, Y., Mesbahi, M., *On the Constrained Attitude Control Problem*, Proceedings of the AIAA Guidance Navigation, and Control Conference, AIAA, Providence, Rhode Island, 2004.

[50] Kim, Y., Mesbahi, M., Singh, G., Hadaegh, F., *On the Convex Parameterization of Constrained Spacecraft Reorientation*, Aerospace and Electronic Systems, IEEE Transactions, Vol. 46, No. 3, pp. 1097-1109, July 2010.

[51] Borchers, B., *CSDP, a C Library for Semidefinite Programming*, Optimization Methods & Software, 11-2(1-4:613-623,1999.

[52] Löfberg, J., *YALMIP: A Matlab Interface to SP, MAXDET and SOCP* Technical report for the Automatic Control Group, Linköping University, LiTH-ISY-R-2328, Jan 2001.

[53] "CLAPACK (f2c'ed version of LAPACK)," Internet, June 2009, Netlib Repository: http://www.netlib.org/clapack/.

[54] Whaley, R., C., Petitet, A., Dongarra, J,. J., *Automated Empirical Optimization of Software and the ATLAS Project,* Oak Ridge National Laboratory, Oak Ridge, TN, 2000.

Vita

Henri Christian Kjellberg $\boxed{\star}$ was born on January 8, 1985 in Vaasa, Finland to Tuija and Anders Kjellberg. He spent his childhood roaming the globe. Henri found his wife, Kristen, in Austin, Texas while working on his undergraduate degree in Aerospace Engineering. Henri is pursuing his Ph.D. in Aerospace Engineering while Kristen is finishing her Doctorate in Veterinary Medicine. They spend their free time racing bicycles and entertaining their dogs and cats.

Permanent address: 2124 Hidden Hollow Cicle Unit A
Bryan, Texas 77807

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.